inf

Ministerul Educației și Învățămîntului
Casa Universitarilor Timișoara

inf nr. 1/88

buletin al CLUBULUI PROGRAMATORILOR

Colectivul de redacție
conf dr ing Crișan Strugaru
ș.l. ing. Ștefan Holban
ș.l. dr. ing. Ionel Jianu
ing. Constantin Cozmiuc
Coperta, tehnoredactarea și prezentarea grafică
ing. Dorin Davideanu

Timișoara iunie 1988

SUMAR

Calculatorul în sprijinul dumneavoastră

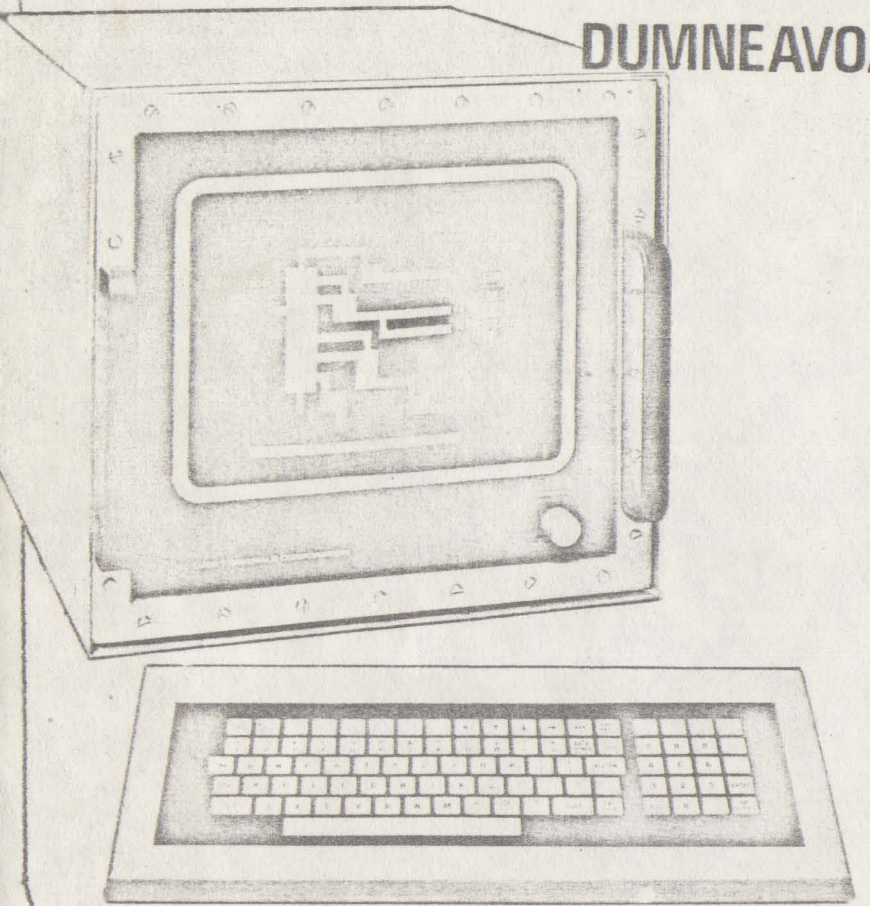
Manuale de utilizare

Programe

Diverse

Calculatorul

ÎN SPRIJINUL DUMNEAVOASTRĂ



Incepînd cu acest număr al buletinului "Clubul programatorilor" editat de Casa Universitarilor din Timişoara, prin străduinţa neobosită a directorului Cornel Secu, Catedra de Automatică şi Calculatoare a Institutului Politehnic "Traian Vuia" Timişoara şi Institutul pentru Tehnică de Calcul şi Informatică (ITCI) împreună cu Fabrica de Memorii Electronice şi Componente pentru Tehnică de calcul (FMECTC) intenţionează să introducă o rubrică permanentă de noutăţi cu referire la evoluţia microcalculatorului TIM-S sub aspectul dezvoltării hardware şi software.

Aflat în producţia de serie ITCI-FMECTC Timişoara din toamna anului 1986, microcalculatorul TIM-S a fost bine primit în masa utilizatorilor de microcalculatoare individuale. Mulţi dintre aceştia, preocupaţi de utilizarea microcalculatorului TIM-S în aplicaţii tot mai complexe, au resimţit pe bună dreptate, absenţa unei memorii externe pe discuri flexibile.

Venind în întîmpinarea acestei dorinţe, ITCI-FMECTC Timişoara, a introdus

recent în fabricaţie de serie un nou echipament EXT-1, prevăzut ca extensie la TIM-S. Livrabil într-o casetă independentă aceasta se conectează la cupla de extensie a microcalculatorului TIM-S şi cuprinde următoarele resurse:

- două unităţi de discuri flexibile;
- interfaţa prin care se realizează interconectarea în reţea locală pînă la a 255 microcalculatoare TIM-S;
- interfaţa pentru comunicaţii în regimul serie (RS232-C) cu rată de transfer programabilă;
- sistemul de operare dedicat să asiste operatorul în utilizarea acestor resurse din BASIC;
- o sursă de alimentare de la care se alimentează şi microcalculatorul TIM-S.

Pentru utilizatorii interesaţi cu deosebire în interconectarea în reţea a microcalculatoarelor TIM-S în versiunea de bază, ITCI-FMECTC oferă şi o soluţie economică. O interfaţă ce cuprinde cîteva imprimante electronice, denumită EXT-2, se conectează la cupla de extensie a microcalculatorului TIM-S. In această versiune subsistemul de operare pentru funcţionarea în reţea se încarcă de pe casetă. Prin interconectarea microcalculatoarelor TIM-S în reţea se deschid perspective interesante privind aplicaţiile în învăţămînt. Cadrul didactic poate avea la dispoziţie un microcalculator TIM-S echipat cu extensia pentru discuri flexibile.

Si în final, o ultimă noutate pentru acest număr al buletinului: a început producţia microcalculatoarelor TIM-S la care tastatura extraplată senzitivă este înlocuită cu o tastatură nouă, ce conţine taste glisante. Sperăm că va satisface cerinţele utilizatorilor.

In numărul următor al buletinului vor apare alte veşti privind dezvoltarea familiei de microcalculatoare TIM-S.

I.P."T.V." Timişoara
I.T.C.Timişoara

CONF. DR. ING. CRIŞAN STRUGARU,
ASIST. ING. CEZAR MORUN

Metode de dialog cu dispozitivele periferice la TIM-S

Sistemul de operare (SO) TIM-S şi extensia sa TIM-S EXT1 sau TIM-S EXT2 au fost proiectate în ideea oferirii unor posibilităţi cît mai mari raportate la dimensiunile lor. Astfel pe de o parte există posibilitatea preluării de către utilizator a oricărei faze din etapele de analiză sintactică, interpretare şi execuţie a unei instrucţii BASIC, pe de altă parte un număr foarte mare de subrutine pot fi apelate, relativ uşor, astfel încît timpul de proiectare şi dimensiunile unui program utilizator să fie cît mai mici.

Un aspect important care apare în exploatarea echipamentelor de calcul este posibilitatea comunicării dintre acestea şi dispozitivele periferice.

Sistemul de operare (SO) TIM-S şi respectiv extensiile sale TIM-S EXT1, TIM-S EXT2 comunică cu dispozitivele periferice ataşate prin:

- instrucţiile IN, respectiv OUT;
- intermediul canalelor de intrare/ieşire (CIO) şi a căilor de transfer de intrare/ieşire (CTIO);
- prin subrutine strict specifice perifericului (SAVE, LOAD, etc.).

1. <u>Dialogul TIM-S - dispozitive periferice prin intermediul instrucţiilor IN, respectiv OUT</u>

Sintaxa acestor instrucţii este:

```
IN     adresă
OUT    adresă, dată
```

şi ele sînt traduse în cod maşină astfel:

```
LD    BC, adresă
IN    A, (C)
```

respectiv

```
LD    A, data
LD    BC, adresă
OUT   (C), A
```

deci adresa are o lungime de 16 biţi (numere cuprinse între o - 65535) iar datele sînt pe 8 biţi.

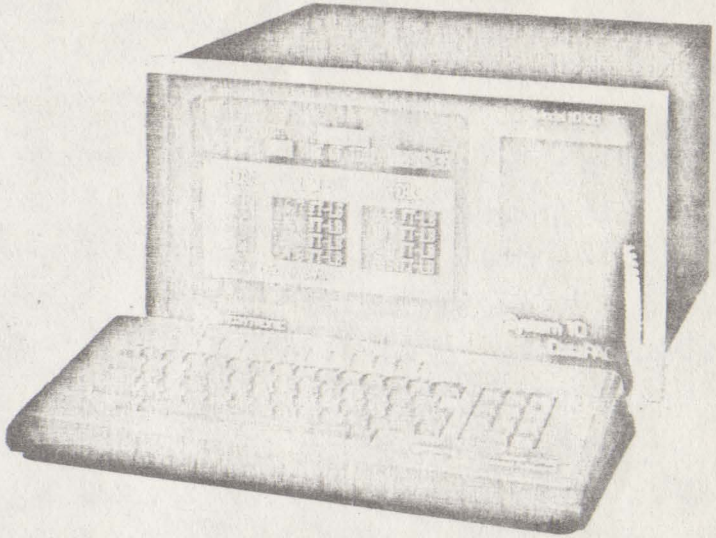
Reamintim că la microprocesorul Z8o în cazul instrucţiilor IN registru, (C) şi OUT (C), registru, pe magistrale de adrese apar:

pe A_{o-7} - conţinutul registrului C

pe A_{8-15}- conţinutul registrului B

Adresele utilizate în TIM-S şi TIM-S EXT1 sînt prezentate în tabelul 1.

Important de reţinut este faptul că A_7 nu se decodifică, şi în tot sistemul de operare, plus programele existente pe piaţa mondială, în cadrul instrucţiilor de I/o A_7=1. Aceasta permite utilizatorului să-şi dezvolte interfeţe şi programe proprii, fără a apare riscul de conflict pe magistrale, folosind A_7=o şi legînd pinul IORQD, de la cupla extensie la A_7.



2. <u>Dialogul prin intermediul canalelor de intrare/ieşire (CIO) şi a căilor de transfer intrare/ieşire (CTIO)</u>

2.1. <u>Prezentare generală</u>

TIM-S poate iniţia un dialog cu 19 periferice logice prin intermediul a 19 CTIO. Acestor periferice logice li se pot ataşa unul din următoarele periferice fizice, prin intermediul unui CIO corespunzător:

- K - tastatură
- S - televizor
- R - zonă de manevră
- P - imprimantă

In cazul ataşării extensiei TIM-S EXT1 sau TIM-S EXT2 mai pot fi ataşate încă 4 periferice şi anume:

- M - disc flexibil
- N - reţea locală
- T - RS-232-C - coduri ASCII
- B - RS-232-C - coduri 8 biţi

Fiecare CIO are un nume indicat mai sus printr-o literă.

Toate imprimantele prezentate în manualul de utilizare TIM-S sînt utilizate prin intermediul CIO-P (canalul de intrare/ieşire de tip P).

Pentru fiecare CIO TIM-S, respectiv CIO-TIM-S EXT1 există o informaţie ataşată, cu următoarea structură.

CIO-TIM-S

- 2 octeţi - Adresă subrutină OUTPUT
- 2 octeţi - Adresă subrutină INPUT
- 1 nume

CIO-TIM-S EXT1 sau TIM-S EXT2

- 2 octeţi - 0008 h
- 2 octeţi - 0008 h
- 1 octet - nume
- 2 octeţi - Adresă subrutină OUTPUT
- 2 octeţi - " " INPUT
- 2 octeţi - Lungime CIO
- n octeţi - Informaţii specifice

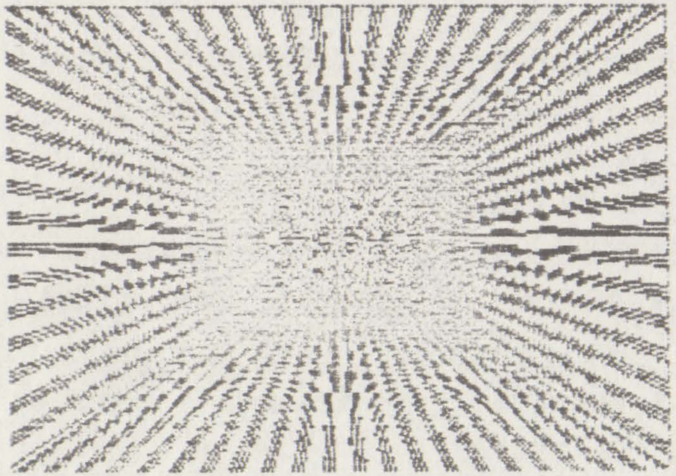
Aceste informaţii se găsesc în RAM începînd cu adresa indicată de variabila CHANS (adresă 23631-2, IY+21, 5C4F-5C50h) şi iniţializarea acestei zone se face conform informaţiei aflate la adresa 15AFh şi prezentată în tabelul 2.

TAB.2

09F4h	- PRINT-OUT
10A8h	- KEY-INPUT
4Bh	- K
09F4h	-
15C4h	- REPORT-D
53h	- S
0F81h	- ADD-CHAR
15C4h	-
52h	- R
09F4h	-

15C4h -
5oh - P
8oh - END MARKER

Perifericele logice şi respectiv căile de transfer I/O ataşate sînt numerotate de la FDh la OFh, primele trei CTIO nefiind accesibile din BASIC.



Pentru fiecare CTIO corespund 2 octeţi într-o tabelă denumită STRMS amplasată de la adresa 235 68 (5CIOh, IY-42) care conţin o adresă relativă la începutul zonei de CIO.

Conţinutul acestei tabele după RESET (punerea sub tensiune a calculatorului) este dat în tabelul 3.

TAB.3

	CIO-adresat		Nr.CTIO
5CIO	ol ooh	K	FD
	o6 ooh	S	FE
	oB ooh	R	FF
	ol ooh	K	oo
	ol ooh	K	ol
	o6 ooh	S	o2
	lo ooh	P	o3
	oo ooh	-	o4-oF

Dacă cei doi octeţi corespunzători unui CTIO sînt oooo înseamnă că acea cale este închisă, şi orice tentativă de a o folosi se va sfîrşi cu un mesaj de eroare.

Deschiderea unui CTIO se face prin:
- RESET, NEW, CLEAR conform tab.3
- prin instrucţia OPEN

Inchiderea unui CTIO se poate face cu:

- CLOSE
- RESET, NEW, CLEAR conform tab.3

De remarcat că instrucţia CLOSE #n n, n=o÷3 va reface cei doi octeţi ataşaţi CTIO corespunzător tab.3.

Tabela de iniţializare a informaţiilor ataşate CTIO se află începînd cu adresa 15C6h.

2.1. Modul de lucru cu CIO şi CTIO din BASIC

Instrucţiile care folosesc această metodă de dialog cu perifericele sînt: PRINT LIST , LPRINT , LLIST , INPUT , INKEY$, OPEN , CLOSE , CLEAR , MOVE.

In continuare se prezintă etapele parcurse de interpreter pentru executarea acestor instrucţii:

2.1.1. OPEN #n, "nume"

a) verificare $o \leq n \leq 15$

b) test nume K,S,R,P,T,B,N,M, etc.

c) se calculează adresa relativă a CIO în zona CHANS şi se introduce în cei doi octeţi corespunzători din STRMS. Observăm că această instrucţie face legătura dintre un periferic logic şi unul fizic.

2.1.2. CLOSE #n

a) verificare $o \leq n \leq 15$

b) se emit informaţiile din buferele de date, în cazul în care la CTIO n este ataşat CIO M, N, sau P.

c) se iniţializează CTIO n conform tab.2

2,1.3. PRINT #n; ...

a) verificare $o \leq n \leq 15$

b) verificare CTIO n deschis

c) CIO ataşat acestui CTIO se face curent prin memorarea adresei sale în variabila de sistem CURCHL (23 633-23 634, IY+23, 5C51-5C52h)

Pentru aceasta se foloseşte secvenţa

```
LD    A,n
CALL  16olh
```

d) în continuare se transmit parametrii din instrucţia PRINT prin intermediul subrutinei de la adresa ooloh (RST lo). Dacă instrucţia PRINT nu se încheie cu ; se transmite suplimentar codul ODh (ENTER).

Parametrii transmişi sînt coduri TIM-S (vezi manualul TIM-S pag.4o-43).

Observaţii

1. Se pot trimite informaţii în mai multe CTIO, în cadrul aceleiaşi instrucţii PRINT.

Exemplu:

PRINT #2;"ECRAN"; #3;"IMPRIMANTA"

2. Subrutina de la adresa ooloh, transmite informaţiile prin intermediul subrutinei OUTPUT a CIO adresat de variabila de sistem CURCHL.

3. Dacă nu se indică numărul CTIO acesta este implicit 2.

4. Aproximativ, în acelaşi mod, se execută şi instrucţiile LIST, LPRINT, LLIST, pentru ultimele două implicit fiind CTIO-3.

5. LLIST #2 va lista prin CTIO-2 şi LIST #3 prin CTIO-3.

6. LLIST, LIST transmit cîte un ODh după fiecare linie de program.

2.1.4. <u>INPUT #n; ...</u>

a) - c) idem instrucţia PRINT

d) informaţiile sînt aduse de la periferic prin subrutina WAIT-INPUT(15DEh), care apelează subrutina INPUT din CIO adresat de CURCHL.

2.1.5. <u>INKEY$#n</u>

Idem, în linii mari, cu INPUT.

Considerăm că această prezentare, fără a mai intra în alte detalii, ne permite trecerea la a analiza modalităţi de dialog cu diferite periferice, din BASIC, folosind instrucţiile prezentate mai sus.

3. <u>Modalităţi de utilizare CIO şi CTIO</u>

3.1. <u>Modificarea unui CIO existent</u>

Utilizatorul are posibilitatea utilizării propriilor subrutine de dialog cu un periferic. Pentru aceasta este suficient să se modifice adresa subrutinelor de OUTPUT, respectiv de INPUT din CIO ales, în zona CHANS. Trebuie manifestată prudenţă în a modifica CIO K,S,R folosite de SO-TIM-S.

3.1.1. <u>Cerinţe impuse subrutinei de INPUT</u>

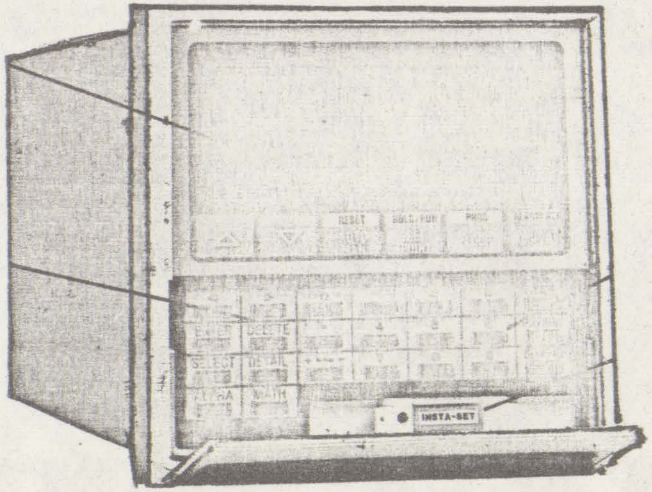
- codul recepţionat în registrul A
- indicatorii de condiţie CY şi Z poziţionaţi astfel:

CY	Z	
1	X	- cod recepţionat
o	1	- se încearcă o nouă recepţie
o	o	- eroare END OF FILE

- regiştrii generali H'L', IY,IX să rămînă neafectaţi.

```
        RET
T-OD    CP      ODh
        RET     NZ - se renunţă la alte
                     coduri de control
                     (TAB,AT etc.)
        CALL    ECA
        LD      A,OAh - LINE FEED (dacă e
                        necesar)
        JR      ECA
ECA-1   ooh          Număr caractere
                     transmise
ECA     PUSH    AF
        CALL    TIP - subrutină de trans-
                      mis cod ASCII la im-
                      primantă (în regis-
                      trul A)
        POP     AF
        CP      OAh
        RET     Z
        CP      ODh
        LD      A,oo
        JR      Z,AECA-1
        LD      A,(ECA-1)
        INC     A
A ECA-1 LD      (ECA-1),A
        CP      n - număr caractere pe
                    rînd
        RET     NZ
        LD      (HL),C
        INC     HL
        LD      (HL),B
        RET
```

Observaţie: Această subrutină poate fi înlocuită cu două instrucţii POKE.



3.1.2. Cerinţe impuse subrutinei de OUTPUT

- codurile transmise sînt coduri TIM-S şi ele trebuiesc traduse în coduri corespunzătoare perifericului respectiv (ASCII etc.)

- registrul A conţine codul transmis
- regiştrii generali H'L',IY,IX să rămînă neafectaţi.

3.1.3. Exemplu

Se doreşte folosirea unei imprimante pentru tipărirea rezultatelor şi listări programe, pe lungime de rînd mai mare de 32 de caractere.

Pentru rezolvarea problemei avem nevoie de două subrutine şi anume:

- INIT - de modificare a CIO-P
- #LPRINT - dialog efectiv

3.1.3.1. Subrutina INIT

```
INIT    LD    HL,(5C4Fh)          - CHANS
        LD    BC,oooFh
        ADD   HL,BC
        LD    BC,adresă  #LPRINT
```

3.1.3.2. Subrutina LPRINT

```
LPRINT   CP    2oh
         JR    C,T-OD
         CP    8oh
         JR    C,ECA - emisie caracter
                       ASCII
         CP    A5h - cel mai mic cod de
                     token
         JP    NC,OB52h - expandare
                          token
         LD    A,ODh
         RST   IOh
         RET
```

Zonele recomandabile de implementare ale acestor subrutine sînt:

a) în primii 16 Ko de RAM, în zona liberă (undeva între adresele 386Eh - 3BFFh, unde se găsesc un număr suficient de locaţii cu conţinut FFh)

b) în buffer-ul de imprimantă (5Booh - 5BFFh)

Atenţie! RESET, NEW modifică informaţia din această zonă.

c) în RAM, în afara zonelor folosite de programul BASIC (cel mai sigur de la adresa FF4oh în jos).

Observaţii:

1. Dacă aceste subrutine se încarcă în primii 16Ko de RAM şi se modifică adresa subrutinei OUTPUT a CIO-P din tabela de la adresa 15AFh, NEW nu va afecta modificările făcute.

2. In situaţia de mai sus, atenţie la protecţia hard la scriere în primii 16Ko de RAM şi la faptul că sistemul trebuie să funcţioneze cu SO-TIM-S din RAM.

3.2. Crearea unui CIO propriu

Această metodă presupune înlocuirea instrucţiei OPEN cu o secvenţă proprie (chiar şi în BASIC) de iniţializare a celor doi octeţi din STRMS corespunzători CTIO dorit şi crearea unui CIO cu subrutine de INPUT şi OUTPUT proprii.

Atenţie!

Instrucţia CLOSE verifică nume CIO şi dă eroare pe alte nume decît cele acceptate de SO.

4. Posibilităţi de utilizare a CIO şi CTIO din programe în cod maşină

4.1. Instrucţia OPEN se poate înlocui cu o scriere directă a deplasamentului faţă de CHANS, a adresei CIO ataşat CTIO, în cei doi octeţi corespunzători din STRMS.

4.2. Iniţializarea variabilei CURCHL se poate face cu secvenţa:

```
LD      A, nr.CTIO
CALL    16o1h
```

4,3. Emiterea unui cod către un CIO făcut curent cu secvenţa de mai sus se face cu

```
LD A, cod
RST 1oh
```

4.4. Recepţia unui cod printr-un CIO făcut curent cu secvenţa de la 4.2. se face în registrul A cu

```
CALL 15DEh
```

4.5. Instrucţia CLOSE se poate înlocui cu refacerea celor doi octeţi din STRMS conform tabelului 2.

5. Posibilitatea transmiterii unor coduri de comandă din BASIC către periferice

SO-TIM-S acceptă pentru instrucţia PRINT (LPRINT) următoarele comenzi TAB,AT, FLASH,BRIGHT,INK,PAPER,OVER, INVERSE.

La analiza sintactică trec iar la execuţie sînt ignorate în CIO-P.

In subrutina noastră de OUTPUT putem prelua aceste coduri şi executa secvenţe de program conform unei convenţii.

Pentru a înţelege metoda, în continuare se arată, în principiu, modul de executare al instrucţiei

```
PRINT #3;AT:2o,1;"A";
LD    A,o3h
CALL  16o1h
LD    A,16h        - cod AT
RST   1oh
LD    A,14h        - ...
```

```
RST  loh
LD   A,olh
RST  loh
LD   A,41h  - cod "A"
RST  loh
RET
```

Reamintim că prin RST loh se ajunge la subrutina de OUTPUT din CIO ataşat CTIO şi făcut curent cu CALL 16olh (adresă memorată în variabila de sistem CURCHL).

După modificarea CIO ales, putem să preluăm şi codurile de control, cărora le dăm funcţii conform unei convenţii.

Exemplu:

TAB	K , L	
	o , m	- avans hîrtie m rînduri
	1 , j	- j spaţii libere
	2 , i	- se alege setul de caractere i

etc.

Alte dezvoltări ale sistemului de operare TIM-S, în acest domeniu, realizate la IPTVT sînt:

- simularea unei memorii FIFO în primii 16 Ko de RAM (deci fără a diminua memoria aflată la dispoziţia utilizatorului în mod uzual);
- memorie de tip LIFO - în aceleaşi condiţii;
- memorie virtuală - în aceleaşi condiţii.

TAB.1

Adr.FIZICA					Adr.LOGICA FOL. IN INTERPRETER	PORT
A4	A3	A2	A1	Ao		
o	o	o	o	o	Eo	PA-8255
o	o	o	1	o	E2	PB-8255
o	o	1	o	o	E4	PC-8255
o	o	1	1	o	E6	PROG-8255
1	o	x	x	o	Fo	ROM/RAM
1	1	x	x	o	FE	PA-8255
1	o	1	1	1	F7	Reţea locală
1	o	1	o	1	F5	"
1	1	1	1	1	FF	Blocare paginare
o	o	o	o	1	E1	Floppy DISC
o	1	o	o	1	E9	"
o	1	o	1	o	E6	"
o	1	o	1	1	EB	"
1	1	o	o	1	F9	"

Observaţii

Adresele A5 - A15 nu sînt folosite în decodificarea adreselor de porturi !

Dialogul TIM-S periferice prin intermediul instrucţiunilor SAVE, LOAD, MERGE, ERASE, CAT, VERIFY va fi tratat în numărul viitor al buletinului.

Bibliografie

1. Manual de utilizare TIM-S
2. Manual de utilizare TIM-S EXT1
3. The Complete Spectrum Rom Disassembly, Ian Logan and Frank O'Hara
4. The Spectrum Operating System, Steve Kramer

ING. PAUL-DAN OPRIŞCĂ

RESOL: un program de rezolvat sisteme mari de ecuaţii liniare

Una dintre metodele cele mai eficiente de rezolvare a sistemelor de ecuaţii liniare cu ajutorul calculatoarelor este folosirea programului specializat RESOL, provenit din biblioteca matematică MATHLIB a calculatorului FELIX C-256.

Bazat pe metoda eliminării Gaussiene şi avînd inclus un algoritm de pivotare a coloanelor pentru a maximiza coeficienţii diagonalei matricii coeficienţilor, RESOL asigură rezolvarea sistemelor într-un interval de timp definit, cu o precizie foarte bună.

Alăturat este prezentată o implementare performantă în BASIC a programului RESOL. Rutina propriu-zisă de rezolvare este cuprinsă între liniile 1oo - 15o. Liniile 3o-6o încarcă datele necesare, iar linia 2oo listează rezultatele obţinute.

Programul RESOL necesită următoarele date de intrare: numărul ecuaţiilor (n), limita minimă (E) sub care un pivot este considerat nul şi deci sistemul este declarat sistem singular, matricea coeficienţilor (A) şi matricea termenilor liberi (B).

Matricea A este o matrice monodimensională, avînd dimensiunea nxm, în care elementele sînt introduse "coloane pe linii". Fie sistemul:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} x \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ b_n \end{bmatrix}$$

Ordinea coeficienţilor în matricea A este: $a_{11}, a_{21}, \ldots, a_{n1}, a_{12}, a_{22}, \ldots, a_{n2}, \ldots, a_{1n}, a_{2n}, \ldots, a_{nn}$

Matricea B este o matrice monodimensională avînd dimensiunea n, ordinea termenilor liberi fiind cea naturală: $b_1, b_2, \ldots, b_n$.

Limita minimă sub care un pivot este considerat nul se ia cu cîteva ordine de mărime sub ordinul de mărime al coeficienţilor matricii A. O valoare foarte mică a acestei limite conduce la nesesizarea singularităţii unui sistem, iar o limită prea ridicată poate declara sistemul singular cînd nu e cazul.

Rutina RESOL transformă matricile A şi B în situ şi nu necesită alte zone de manevră, ceea ce permite rezolvarea sistemelor foarte mari. Pentru o memorie RAM de 48 Ko limita memoriei este atinsă de un sistem de aproape 9o de ecuaţii. La sfîrşitul execuţiei matricea A este distrusă, iar matricea B conţine soluţia sistemului.

Una din metodele care a dus la accelerarea execuţiei în BASIC cu cca 2o% a fost utilizarea numelor de variabile dintr-o singură literă. Sînt folosite toate literele cu excepţia A, B care sînt matrici şi i, j

rămîne la dispoziția utilizatorului pentru contoare etc.

Implementarea prezentată poate funcționa ca atare sau poate fi compilată atît cu compilatorul SOFTER, varianta FULL, cît și cu compilatorul BLAST. In cazul compilării cu compilatorul SOFTER, durata compilării este foarte scurtă, însă acesta nu acceptă liniile 4o și 5o, astfel că trebuie găsită o altă metodă de introducere a datelor. De asemenea, el nu produce un program independent. Aceste dezavantaje sînt eliminate la compilarea cu BLAST.

In cazul compilării cu BLAST, pentru a mări viteza de execuție se adaugă instrucțiunea: 2o REM ! INT c,d,g,h,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z declarînd variabilele respective întregi.

Testele s-au făcut pe un sistem de 24 de ecuații. Execuția rutinei RESOL în BASIC a durat 255 sec., a codului generat de compilatorul SOFTEK 7o sec., iar a codului generat de compilatorul BLAST 4o sec. Rezolvarea unui sistem de 8o de ecuații a fost făcută de codul generat de BLAST în 25 de minute.

```
  1o REM SISTEM
  2o REM ! INT c,d,g,h,k,l,m,n,o,p,q,s,t,u,
v,w,x,y,z
  3o LET a$ ="Număr ecuații:  ": INPUT (a$ )
:n: PRINT  a$ :n: LET a$ ="Pivot minim: ":
INPUT (a$ ):E:PRINT ´a$ :E: LET b$ ="Fisier
matrice ": PRINT ´b$ :"A: ":: INPUT (b$ ):
"A: ":a$ : PRINT a$ : PRINT ´b$ :"B: ":: IN
PUT (b$ ):"B: ":b$ : PRINT b$ : PRINT #0:"
Start banda pentru citire A,B.": DIM A(n n):
 DIM B(n)
  4o LOAD a$ DATA A()
  5o LOAD b$ DATA B()
  6o CLS : PRINT AT 0,0:"RESOL"
 1oo REM RESOL
  11o LET c=-n: FOR h=1 TO n: PRINT AT 0,6:h:
LET d=h+1: LET c=c+ n+1: LET F=0: LET t=c-h:
FOR g=h TO n: LET p=t+g: IF ABS F< ABS A (p)
THEN LET F=A(p): LET s=g
  12o NEXT g: IF ABS F<E THEN PRINT ´"SISTEM
SINGULAR": STOP
  13o LET q=h+n *(h-2): LET t=s-h:
  FOR k=h TO n: LET q=q+n: LET p=q+t: LET R=A
(q): LET A(q)=A(p): LET A(p)=R: LET A(q)/F:
NEX T k: LET R=B(s)=B(h): LET B(h)=R/F: IF h
=n THEN GO TO 15o
```

```
14o LET o=n  (h-1): FOR l=d TO n: LET u=o+l:
FOR m=d TO n: LET v=n  (m-1)+1: LET w=v+t:
LET A(v)=A(v)-A(u) * A(w): NEXT m: LET B(l)=
B(l)-B(h) * A(u): NEXT l: NEXT h
 15o LET t=n  n: FOR h=1 TO n-1: LET x=t-h:
LET v=n-h: LET z=n: FOR k=1 TO h: LET B(y)=
B(y)-A(x) * B(z): LET x=x-n: LET z=z-1: NEXT
k: NEXT h
 2oo FOR h=1 TO n: PRINT "x";h;"= ";B(h):
NEXT h
```

ING. MIRCEA POPA

Conectarea imprimantei Robotron 1157 la microcalculatorul TIM-S

1. Introducere

In scopul creşterii performanţelor calculatorului TIM-S, acesta a fost prevăzut cu o interfaţă paralelă şi una pentru conectarea unor imprimante cu tipuri de dialog diferit. Imprimantele care au fost conectate şi modul de comandă a lor utilizînd instrucţiuni din limbajul BASIC sînt prezentate în manualul de funcţionare şi utilizare a microcalculatorului TIM-S.

Dată fiind răspîndirea relativ largă a imprimantei ROBOTRON 1157 s-a pus problema conectării ei la microcalculatorul TIM-S. Lucrarea prezintă un mod de rezolvare a acestei probleme.

2. <u>Interfaţa cu imprimanta ROBOTRON 1157</u>

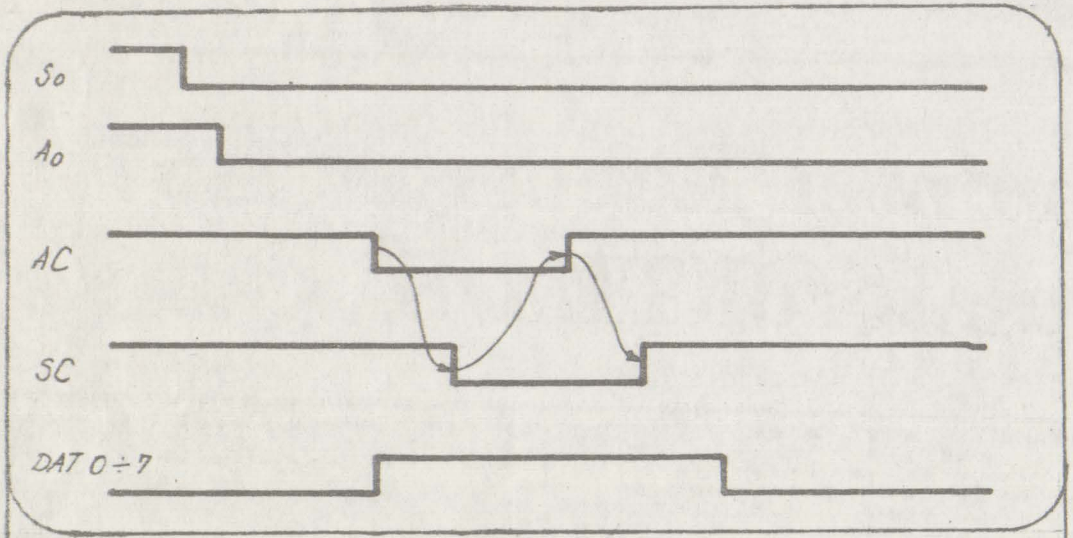


fig.1

Figura 1 prezintă cronograma semnalelor pentru transferul unui caracter, semnificaţia lor fiind următoarea:

S_o: activ la "0" - indică că emiţătorul va face o transmisie;

A_o: activ la "0" - indică că receptorul este pregătit pentru imprimare;

AC: activ la "0" - indică că receptorul poate să preia un caracter;

SC: activ la "0" - semnal de strob pentru date;

DAT0.7 - sînt linii de date - datele se transmit în cod ASCII şi complementate.

Pentru conectarea acestei imprimante la microcalculatorul TIM-S se utilizează interfaţa paralelă existentă deci nu sînt necesare circuite suplimentare. La cupla paralelă se fac următoarele conexiuni:

semnal	cuplă
DAT 0 - 6	pinii 1 - 7
SC	pinul 8
AC	pinul 9
S_o şi masa	pinii 14 - 22

Se remarcă că semnalul S_o este permanent conectat la masă iar semnalul A0 nu este testat presupunînd că imprimanta este pregătită pentru imprimare.

Pentru comanda imprimantei a fost concepută subrutina pentru transmisia unui caracter, respectînd cromograma din figura 1, dată mai jos:

```
ET1     LD      C,A
        CALL    1F54
        JP      NC,3AFB
        IN      A,(FE)
        RLA
        JR      NC,ET1
        LD      A,C
        CPL
        AND     7F
        OUT     (E2),A
        OR      8o
        OUT     (E2),A
ET2     CALL    1F54
        JP      NC,3AFB
        IN      A,(FE)
        RLA
        JR      C,ET2
        LD      A,C
        CPL
        AND     7F
        OUT     (E2),A
        LD      A,00
        OUT     (E2),A
        RET
```

Subrutina primeşte în registrul A codul ASCII complementat al caracterului de tipărit. Prima instrucţiune salvează acest cod în registrul C întrucît registrul A va fi folosit în continuare. Următoarele cinci instrucţiuni verifică dacă linia AC a devenit activă, indicînd o cerere din partea imprimantei. Dacă programul se blochează în această buclă, din cauza apariţiei unei defecţiuni se poate ieşi din această stare prin acţionarea tastei BREAK. In continuare se readuce codul caracterului de tipărit în registrul A şi se plasează pe liniile DATO 6 activînd şi semnalul SC. Apoi se verifică dacă linia AC a devenit inactivă, existînd şi aici posibilitatea de a părăsi această buclă prin acţionarea tastei BREAK. Dacă semnalul AC este inactiv se dezactivează semnalul SC şi apoi se dezactivează şi liniile de date.

Pentru realizarea legăturii între această subrutină şi comenzile din limbajul BASIC există trei posibilităţi:

1. Plasarea subrutinei la adresa 3BCO, unde începe o zonă neutilizată din memoria EPROM şi modificarea conţinutului loca-

ţiilor 38D4 - 38D6 conform celor de mai jos:

adresă	conţinut nou
38D4	C3
38D5	C0
38D6	3B

Aceasta înseamnă că utilizatorul va avea la dispoziţie comenzile din limbajul BASIC referitoare la tipărirea în regim paralel pentru imprimantele CDC 9335 şi ROBOTRON K6311, un acelaşi microcalculator neputînd să comande şi imprimanta ROBOTRON 1157 şi imprimantele CDC 9335 sau ROBOTRON K6311. Acest dezavantaj apare datorită faptului că nu există suficient spaţiu neutilizat în memoria EPROM pentru creerea unor noi comenzi aferente tipăririi cu imprimanta ROBOTRON 1157.

Pentru realizarea modificărilor prezentate şi înscrierea subrutinei în memorie există două posibilităţi:

a) reprogramarea memoriei EPROM la adresele 38D4 - 38D6 şi 3BC0 - 3BE1;

b) ţinînd seama de faptul că sistemul de operare al microcalculatorului TIM-S este operaţional în memoria RAM se poate proceda în felul următor:

- se încarcă programul utilitar "MONS3M";

- cu acest program se înscrie în memoria RAM utilizator programul:

```
DI
LD    A, FB
OUT   (E4), A
EI
JP    (Adresă + 29)
```

unde "Adresă" este adresa la care a fost încărcat programul utilitar, şi apoi se lansează în execuţie acest program;

- se fac modificările de la adresele 38D4 - 38D6 şi 3BC0 - 3BE1 folosind tot programul utilitar;

- se înscrie în memoria RAM utilitar programul:

```
DI
LD        A, F9
OUT       (E4), A
EI
JP        (Adresă + 29)
```

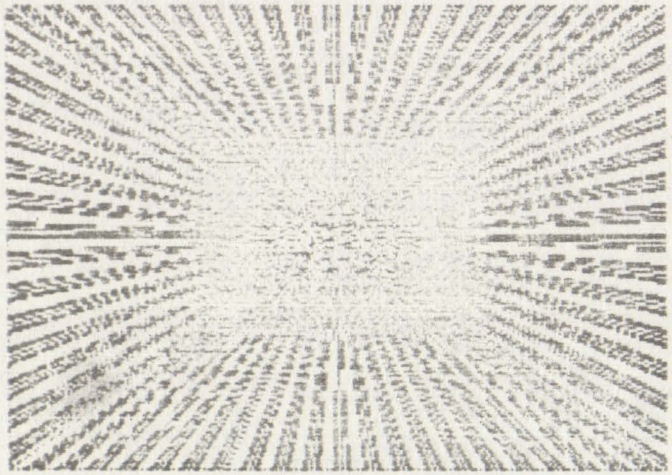
şi se lansează în execuţie;

- se revine în interpretorul BASIC.

După realizarea celor prezentate

mai sus se poate comanda imprimanta ROBOTRON 1157 cu comenzi din limbajul BASIC.

2. Plasarea subrutinei de transfer a unui caracter în spaţiul RAM utilizator. In acest caz utilizatorul va trebui să conceapă şi programul care foloseşte această subrutină, lansarea sa în execuţie făcîndu-se cu comanda RANDOMIZE USR Adresă. Această soluţie nu afectează memoria EPROM dar reduce capacitatea spaţiului RAM utilizator.



3. Plasarea subrutinei de transfer şi a programului care o foloseşte în tamponul pentru imprimare, zonă de memorie RAM de 256 octeţi, începînd cu adresa 5BOOH. In acest caz trebuie realizată următoarea modificare: la adresa indicată de variabila de sistem CHANS (5C4FH şi 5C50H) se adună valoarea OFH şi se obţine o nouă adresă unde se plasează OOH iar la adresa următoare 5BH, 5BOOH fiind adresa de început a programului care realizează tipărirea. Această soluţie nu afectează memoria EPROM, nici memoria RAM utilizator dar va fi necesară evitarea acelor comenzi din limbajul BASIC care folosesc zona 5BOOH-5BFFH ca tampon pentru imprimare.

3. Concluzii

Soluţia descrisă asigură conectarea imprimantei ROBOTRON 1157 la microcalculatorul TIM-S. Pentru comanda imprimantei, de către utilizator, trebuie să se ţină seama de adresa unde a fost plasată subrutina de transfer, conform cu cele prezentate mai sus.

4. Bibliografie

1. * * * - Documentaţia tehnică a microcalculatorului TIM-S
2. * * * - Documentaţia tehnică a imprimantei ROBOTRON 1157
3. * * * - Microcalculator TIM-S - Manual de funcţionare şi utilizare
4. * * * - INF nr.1-2/1987

MIRCEA TEODORESCU,
LAURENȚIU EMIL

Creion optic & Kempston joystick

CUPRINS: 1Ce este un creion optic ?
2 Cum se conecteaza creionul optic ?
3 Utilizarea creionului optic.
4 Creionul optic în funcția de selecție.
5 Desenarea cu creionul optic.
6 Utilizarea programului cod-mașină.
7 Condițiile de apariție ale erorilor.
8 Schema creionului optic & Kempston J.

1 Ce este un creion optic ?

Creionul optic este un dispozitiv care detectează lumina emisă de un ecran de televizor și transmite semnale calculatorului funcție de poziția creionului pe ecran. Cînd este utilizat cu un software adecvat, poate fi utilizat la alegerea unei funcții de pe ecran pentru a fi pusă în execuție, sau poate fi folosit la desenarea unor proiecte pe televizor care eventual pot fi salvate apoi pe banda, imprimantă, etc.

Funcționează în modul următor: Calculatorul generează două linii pe ecran, una verticală și una orizontală pe care le baleiază pe ecran cu viteză foarte mare, în momentul în care se pune creionul pe ecran, creionul sesizează dreptele cînd trec prin dreptul lui și transmite un semnal calculatorului care aproximează poziția creionului grossiere apoi o calculează cu precizie stabilind cu o aproximație bună poziția creionului pe ecran.

O serie de programe în cod-mașină sînt prevăzute pentru a permite desenarea de cercuri, linii, dreptunghiuri, arcuri de cerc, și pentru a umple contururi cu oricare din cele 8 culori ale calculatorului. Pentru detalii se studiază secțiunea 5.

2 Cum se conectează creionul optic la calculator

Creionul optic se conectează la calculator prin intermadiul unei interfețe a cărei schemă o prezentăm în secțiunea 8 a articolului.

Se procedează în felul următor:
(a) se conectează creionul la interfață;
(b) se asigură dacă este nealimentat calculatorul și se introduce interfața în calculator (proiectată pentru HC-85 & Spectrum 48K);
(c) se alimentează calculatorul și se încarcă programul de pe casetă.

Programul are două părți: BASIC & Cod-mașină.

3 Utilizarea creionului optic

Creionul trebuie ţinut perpendicular pe ecran pentru a asigura o luminare optimă şi pentru a opri reflexiile care ar afecta o bună operare cu ajutorul lui.

Se poate observa o uşoară nepotrivire între poziţia ţintei relativ la poziţia vîrfului creionului pe ecranul TV-ului. Aceasta se poate remedia prin reglarea strălucirii, contrastului şi a coloritului TV-ului sau se poate varia poziţia ţintei faţă de vîrful creionului apăsînd tastele I şi Q aşa cum se explică în programul de calibrat. După calibrare, programul astfel modificat poate fi salvat pe bandă, numai partea de cod-maşină (de preferat pe o casetă nouă).

În operarea cu creionul optic există două tipuri de operaţii :

(a) dacă se utilizează creionul în selecţia dintr-o listă de funcţii, se pune creionul pe varianta dorită ;

(b) dacă se utilizează creionul pentru desenare, atunci se punctează pe ecran funcţia dorită apoi se apasă o tastă oarecare.

4 Creionul optic în funcţia de selecţie

Un program cod maşină facilitează creionul optic în operaţia de căutare şi execuţie a unei funcţii dintr-o listă. Programul afişează numărul liniei variantei alese, de aceea este necesar ca fiecare variantă să fie pusă în listă pe linii distincte. Varianta aleasă poate fi determinată prin utilizarea numărului liniei afişate de programul cod-maşină. Spre exemplu :

```
VARIANTA 1   (line 0)
VARIANTA 2   (line 1)
VARIANTA 3   (line 2)
```

Programul cod-maşină utilizat "MENU" se găseşte la adresa 63109 şi este folosit astfel :

```
LET Lno=USR 63109
```

Cînd se întoarce din cod-maşină, Lno conţine numărul liniei la care se găseşte creionul şi poate fi testată-n BASIC astfel :

```
IF Lno=0 THEN GO SUB VARIANTA 1
IF Lno=1 THEN GO SUB VARIANTA 2
IF Lno=2 THEN GO SUB VARIANTA 3
```

sau poate fi utilizată forma GO SUB Lno*100+200

Programul cod maşină întoarce foarte repede numărul liniei la care se găseşte creionul, de aceea trebuie inserat un program de întîrziere:

```
1000 LET X=in 63:IF X/2<>INT(X/2) THEN GOTO 1000
1010 PAUSE 25: REM întîrziere pentru a permite poziţionarea corectă a creionului
1020 LET Lno=USR 63109
```

Dacă se doreşte salvarea programului cod-maşină care permite facilitatea descrisă, separat de programul cod maşină principal se procedează astfel:

1. Se încarcă "LP 48 v5.0" prin:
```
CLEAR 59059
LOAD "LP 48 v5.0"CODE
```
2. Se salvează programul "MENU" :
```
SAVE "MENU" CODE 63030,96
```
3. pentru reîncărcare se tastează:
```
CLEAR 63029
LOAD "MENU" CODE
```

5. Desenarea cu creionul optic

Programul "LP 48 v5.0" conţine 16 funcţii principale care pot fi selectate dintr-o listă care se găseşte pe ultimele două linii în partea de jos a ecranului.

Lista arată astfel:

```
E D M C R F H B I P N T K F A L
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
```

Literele sînt iniţialele unor funcţii care se apelează astfel:

1. Se punctează creionul pe pătratul corespunzător literei;
2. Se apasă o tastă oarecare.

Funcţia dorită va fi astfel executată.

Programul utilizează două variabile numite "ORIGINEA" care este indicată pe ecran prin "X" şi "TINTA" care este reprezentată pe ecran prin două axe de coordonate ortogonale. Poziţia acestor simboluri determină capetele liniilor colţurile dreptunghiurilor, centrul şi circumferinţa cercurilor.

Descriem mai jos funcţiile în detaliu:

ERASE

Această comandă prmite ştergerea liniei sau figurii(dreptunghi, cerc sau arc de cerc) care tocmai a fost desenată. Dacă se schimbă poziţia Tintei sau Originii atunci figurile nu mai pot fi şterse.

Se apelează prin punctarea literei E şi apăsarea unei taste.

DRAW

Această funcţie desenează o linie din Origine în Tintă. sînt necesari trei paşi:

(a) Se fixează unul din capetele dreptei prin poziţionarea Originii pe locul dorit(folosind comanda MOVE)

(b) Se fixează celălalt capăt al dreptei prin plasarea creionului pe locul dorit şi apăsarea unei taste (se deplasează astfel Tinta)

(c) Se pune creionul pe litera D şi se apasă o tastă oarecare

MOVE

Această comandă deplasează Originea peste poziţia Timtei. Este utilizată în comenzile DRAW, CIRCLE, RECTANGLE & ARC pentru a defini unul din puncte.

Se pune creionul pe M şi se apasă o tastă.

CIRCLE

Aceasta se foloseşte tot în trei paşi pentru a trasa un cerc:

(a) Se fixează centrul cercului prin poziţionarea originii (X) ;

(b) Se fixează raza cercului prin deplasarea ţintei;

(c) Se pune creionul pe C şi se apasă o tastă.

RECTANGLE

(a) Se deplasează Originea pe unul din colţurile dreptunghiului care va fi desenat;

(b) Se pune Ţinta pe colţul diagonal opus;

(c) Se pune creionul pe R şi se apasă o tastă.

FILL

Această funcţie permite umplerea unui contur închis cu una din cele 8 culori ale calculatorului.

(a) Se plasează Ţinta în interiorul conturului care va fi umplut (colorat);

(b) Dacă este necesar se foloseşte comanda INK pentru a alege culoarea dorită ;

(c) Se pune creionul pe F şi se apasă o tastă.

HAND-DRAW

Este folosită pentru a desena cu mîna liberă - creionul desenează în funcţie de deplasarea lui pe ecran. Mişcarea trebuie să fie lentă, pentru a permite calculatorului urmărirea creionului pe ecran. Se punctează litera H şi se apasă o tastă. Se pune creionul pe ecran de unde se doreşte începerea schiţei apoi se apasă o tastă pentru a începe desenarea.

Pentru a opri desenarea se apasă o tastă.

BORDER, INK, PAPER

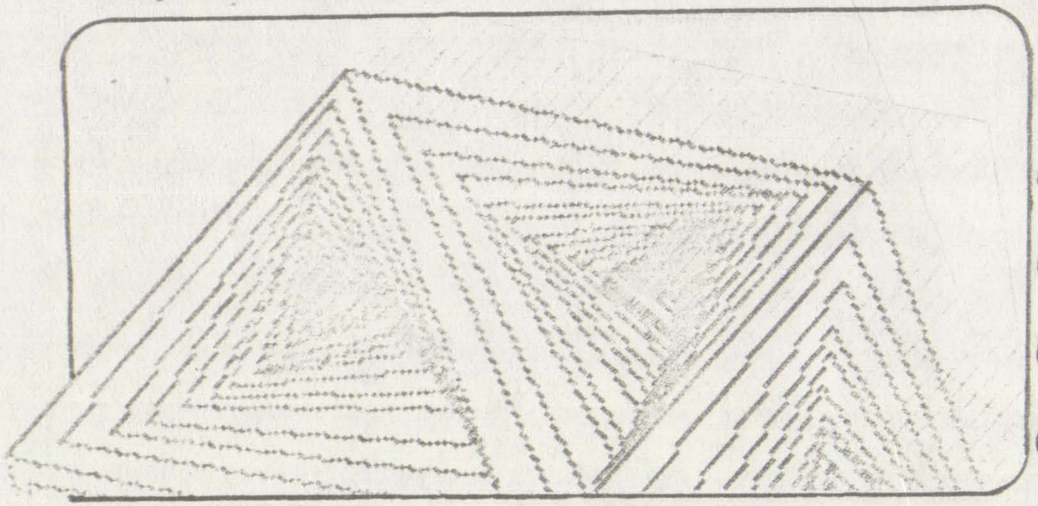
Aceste trei comenzi lucrează similar. Ele permit schimbarea culorii, borderului, hîrtiei şi cernelei, cum de altfel sugerează şi titlul lor. Trebuie reţinut că schimbarea culorii hîrtiei se va face numai după ştergerea ecranului prin funcţia NEW SCREEN.

NEW SCREEN

Această comandă permite ştergerea ecranului pentru a se porni din nou cu un alt desen. Va curăţa ecranul şi va schimba culoarea ecranului funcţie de hîrtia şi cerneala curentă care pot să difere de ecranul anterior. Originea şi Ţinta sînt setate la centrul ecranului.

TAPE

Se foloseşte pentru salvarea sau încărcarea de pe casetă a pozelor desenate. Pentru a ieşi din această funcţie se punctează pe ABORT şi se apasă o tastă.



KEEP

Această funcţie copiază imaginea ecran în memoria calculatorului. Imaginile ecran sînt introduse deasupra spaţiului alocat BASIC-ului şi vor rămîne în siguranţă chiar şi la comanda NEW. Pot fi introduse cel mult 5 ecrane după ce s-a executat comanda NEW (altfel spaţiul este de numai 4 ecrane).

Pentru utilizare se punctează litera K şi se apasă o tastă la întîmplare.

RECALL

Această comandă permite rechemarea instantanee a copiilor imaginilor ecranelor care au fost stocate în memorie (vezi comanda KEEP).

Dacă mai mult de un ecran a fost reţinut, opţiunea de a selecta una dintre pagini sau ciclarea paginilor reţinute poate fi făcută. Intîrzierea dintre afişarea paginilor poate fi variată funcţie de preferinţa programatorului. Intîrzierea dorită se obţine prin punctarea cu creionul a numărului dorit şi apăsarea unei taste.

Scala modificării întîrzierilor poate fi modificată prin POKE-uri la adresa 61296 cu numere între 1 şi 5.

ARC

Trei puncte sînt necesare pentru definirea unui arc de cerc :

(a) "Originea" defineşte un capăt al arcului;
(b) "Ţinta" defineşte celălalt capăt al arcului
(c) Poziţia intermediară a Ţintei determină curbura arcului.

Sînt necesari patru paşi în desenarea arcului

1. Se deplasează Originea (X) acolo unde va fi unul din capetele curbei;
2. Se defineşte curbura arcului prin pziţionarea Ţintei (+) ;
3. Se punctează locul unde va fi celălalt capăt al arcului şi se deplasează Ţinta. (Aceasta nu va afecta poziţia intermediară fixată anterior, în pasul 2);
4. Se pune creionul pe litera A şi se apasă o tastă.

LETTERS

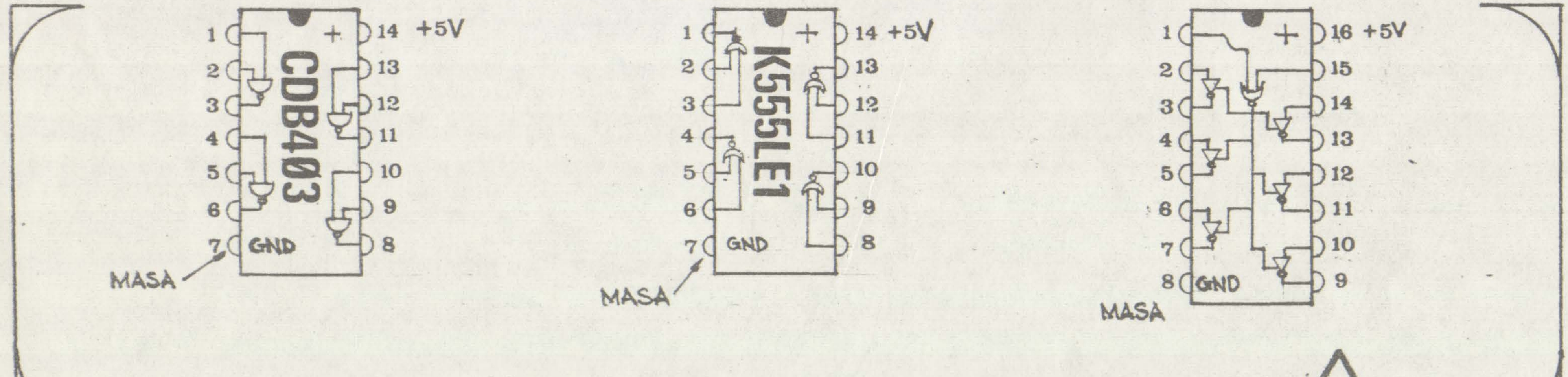
Această comandă permite inserarea de texte pe ecran - pot fi scrise numere, litere sau chiar caractere grafice.

Textul va fi plasat începînd din poziţia curentă a Ţintei şi sînt necesari 3 paşi :
(a) Poziţionarea Ţintei pe locul din care vom insera textul dorit;
(b) Se punctează litera L şi se apasă o tastă;
(c) Se tastează textul dorit, apoi se apasă tasta ENTER şi textul este introdus pe ecran încheindu-se astfel cu comanda LETTERS.

6. UTILIZAREA PROGRAMULUI COD-MASINA CA PROGRAM INDIVIDUAL

Programul cod-maşină care urmează după programul de BASIC poate fi folosit individual fără a mai încărca şi partea de BASIC. Odată familiarizat cu creionul optic se va observa inconvenientul încărcării şi salvării separate a codului maşină de aceea se va tasta:
CLEAR 59059
LOAD"" CODE şi se va încărca de pe casetă partea de cod normal.

Pentru salvarea acesteia se tastează:
SAVE "LP 48 v5.0" CODE 59060,5220
Sînt trei metode de a porni programul cod maşină:
(a) cu calibrare urmată de
(b) iniţializare a culorilor (BORDER, INK şi PAPER) şi setarea variabilelor la valorile de Start şi
(c) fără iniţializare, astfel încît orice ecran memorat poate fi accesat.
Adresele sînt după cum urmează:
Pentru (a) se tastează:
RANDOMIZE USR 59478.
Pentru (b) se tastează:
RANDOMIZE USR 59481
Pentru (c) se tastează:
RANDOMIZE USR 59487
Pentru distrugerea ecranelor reţinute în memorie se foloseşte CLEAR 59477 ; re-START-ul se face ca la (b).



K155LN6 (6 BUFFERE TRI-STATE INVERSOARE) △

Este posibilă utilizarea creionului optic în programe proprii astfel încît programul cod-maşină întoarce coordonatele X şi Y ale punctului indicat de creion (cu o precizie relativă) Aceasta se poate face astfel:

LET coords= USR 62 564

iar apoi coordonata Y se calculează prin:

LET Y = INT (coords/256)

şi coordonata X:

LET X = coords-256*Y

Valorile rezultate pentru X şi Y pot fi folosite în comenzi ca: PLOT, DRAW, CIRCLE, sau POINT. Pentru a salva această rutină se tastează:

SAVE "hpen" CODE 62564,1706

şi reîncărcarea:

CLEAR 62563

LOAD "hpen" CODE

7. CONDITIILE DE APARITIE ALE ERORILOR

Anumite operaţii pot genera erori, din care unele vor cauza întoarcerea în BASIC iar altele nu. Dacă se produce întoarcerea în BASIC programul poate fi repornit din nou cu ajutorul adreselor de re-START fără pericolul pierderii ecranelor memorate- Adresa de la punctul (c).

Pin	A	B
28	Necuplat	Adresa 11
27	Adresa 1Ø	Adresa 9
26	Adresa 8	BUS ACK.
25	Refresh	ROM CHIP SELECT
24	MACHINE 1	Adresa 4
23	-12V	Adresa 5
22	+12V	Adresa 6
21	WAIT	Adresa 7
20	-5V	RESET
19	WRITE	PASS Request
18	READ	U Sincro Cadre+Linii
17	I/O Request	V Sincro Linii
16	Mem Request	Y Sincro Cadre
15	HALT	VIDEO
14	Non Mask. Inter.	MASĂ
13	Interrupt	I.O.R.G.E.
12	DATA 4	Adresa 3
11	DATA 3	Adresa 2
10	DATA 5	Adresa 1
9	DATA 6	Adresa Ø
8	DATA 2	CLOCK nu
7	DATA 1	MASĂ
6	DATA Ø	MASĂ
5	SLOT (cheia)	(cheia) SLOT
4	Necuplat	+9V
3	DATA 7	+5V
2	Adresa 13	Adresa 12
1	Adresa 15	Adresa 14

MAGISTRALA DE DATE (VEDERE DIN SPATE SPRE FAŢA)

Unele din erorile posibile sînt:

CIRCLE - Dacă se încearcă desenarea unui cerc prea mare, care depăşeşte ecranul.

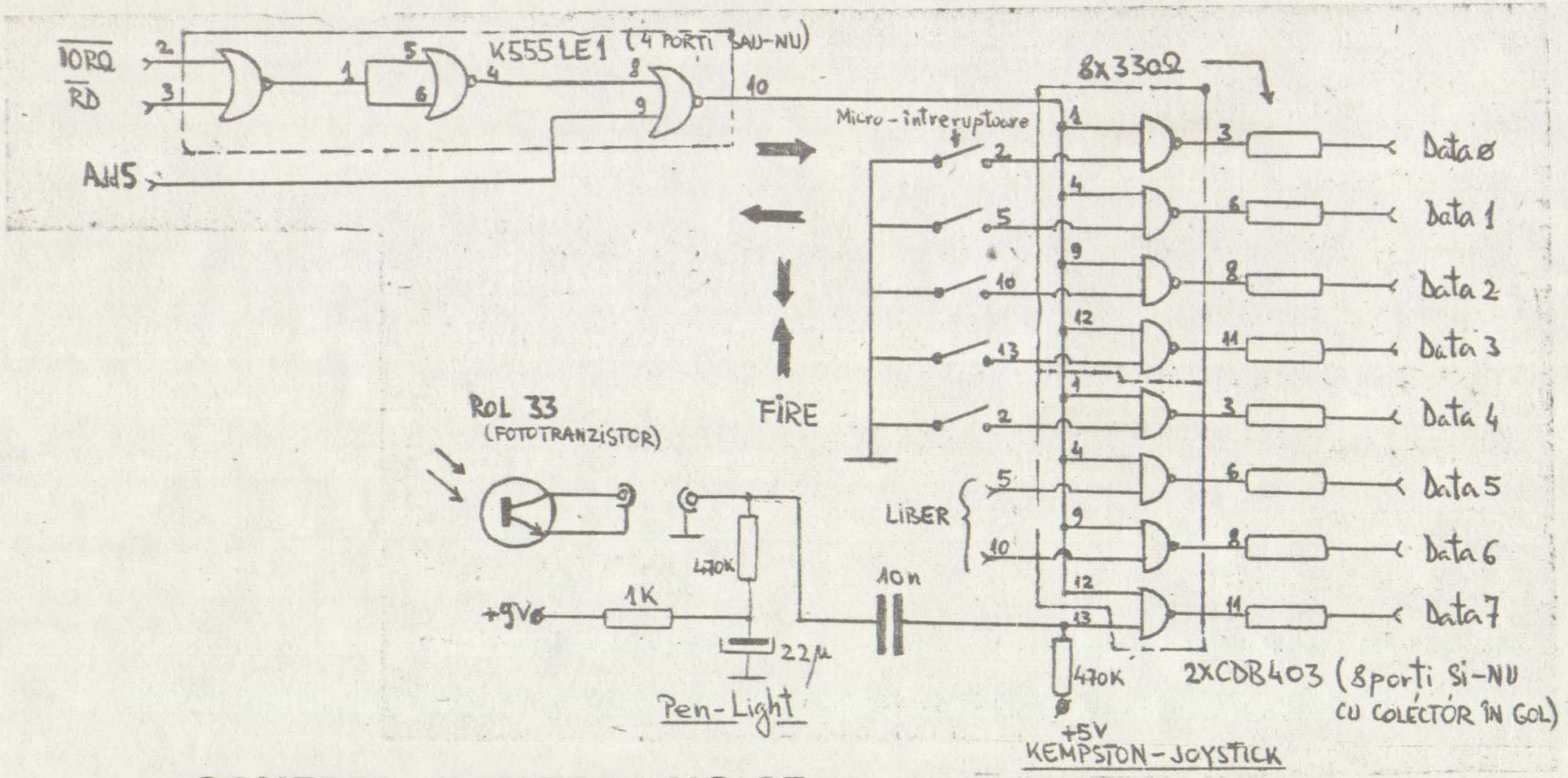
KEEP - Dacă nu mai este spaţiu pentru reţinerea în memorie a ecranului (fiecărui ecran i se alocă un spaţiu de aproape 7K-6912).

RECALL - Dacă se încearcă rechemarea unui ecran din memorie înainte de a fi reţinut.

ARC - Dacă arcul de cerc depăşeşte ecranul.

TAPE - Erorile standard ale BASIC-ului vor apare dacă apar erori la încărcarea de pe bandă. In acest caz se procedează astfel: RANDOMIZE USR 59847 pentru a se reporni programul.

<u>DIFICULTATI IN CALIBRARE</u>: Dacă nu se va realiza calibrarea corectă a creionului se va experimenta cu factori de timp care ar putea îmbunătăţi acest lucru. Adresa este: 63056
Valoarea este: 13



SCHEMA PENTRU HC-85

8. SCHEMA CREIONULUI OPTIC & KEPSTON JOYSTICK

Deoarece programul :"LP 48 v5.0" nu coincide cu programul "Lpmc" care funcţiona pe o altă schemă care se conecta la intrarea de casetofon a calculatorului, precizăm ce schimbări trebuiesc făcute în programul "Lpmc" pentru a se asigura funcţionarea pe schema nouă care o prezentăm mai departe. După încărcarea programului "Lpmc" se iese în BASIC şi se tastează :

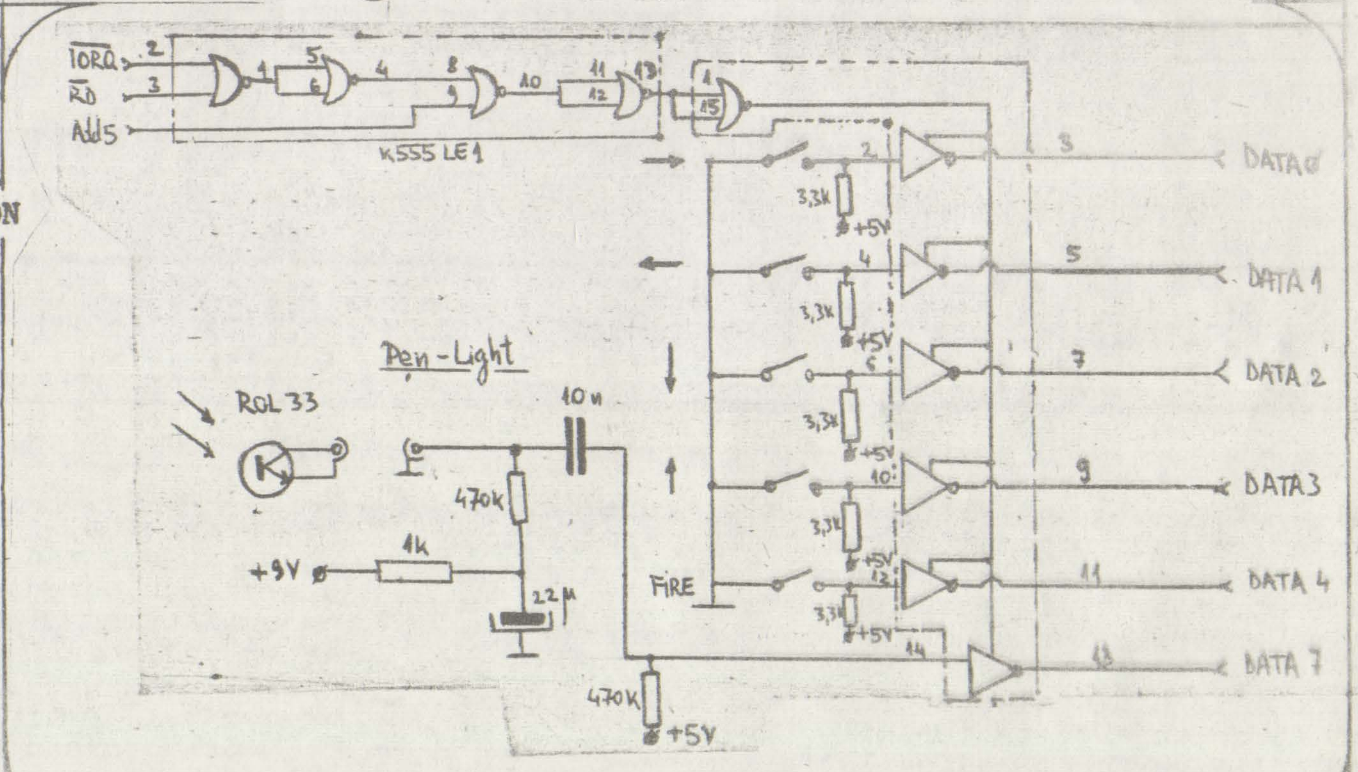
```
POKE 63031,223
POKE 63052,120
POKE 63053,32
```

apoi se porneşte programul cu adresele date în secţiunea 6.

Prima schemă funcţionează pentru un calculator HC 85 şi conţine schemele pentru KEMPSTON şi creion optic, iar a doua schemă funcţionează pe calculatoarele SPECTRUM 48K, avînd în componenţă aceleaşi funcţii ca mai sus. Conectarea la calculator se face prin magistrala de date care se găseşte în partea din spate a calculatoarelor amintite (pot merge şi pe TIM-S cu mici modificări neesenţiale). Circuitele integrate sînt : CDB 403
K555 LE1 (URSS)
LN6 (URSS)

Joystickul poate fi făcut cu ajutorul a patru micro-întreruptoare pentru deplasări şi un al cincilea buton care poate fi eventual o tastă de terminal pentru FIRE.

Dispunerea micro-întreruptoarelor se va face în cruce pentru a se putea executa patru comenzi directe plus 4 diagonale, deci 8 deplasări. Desigur că Joystickul poate fi proiectat şi altfel. Pe lîngă cele două scheme prezentăm şi schema magistralei de date care se găseşte la spatele calculatorului (HC 85 şi SPECTRUM 48K).



SCHEMA PENTRU SPECTRUM 48K

ASIST. DR. ING. MIRCEA STRATULAT, ASIST. ING. MARIUS CRIȘAN,
ING. CONSTANTIN COZMIUC

Microcalculatorul TIM-S în achiziția, prelucrarea și distribuția datelor fizico-chimice

Eficienţa activităţii de cercetare,în laborator, a unor procese de natură fizico-chimică poate fi crescută considerabil,la un preţ de cost scăzut,prin utilizarea unui microcalculator de tip personal şi a unui sistem de interfaţă cu procesul. Prin această structură,care prezintă funcţiile unui calculator de proces,se pot urmări şi comanda toate mărimile fizice de interes, pe întreaga durată de evoluţie a procesului.Adaptarea caracteristicilor informaţiilor din proces la cele ale informaţiilor ce pot fi introduse în calculator,precum şi a caracteristicilor informaţiilor produse de calculator,la cele ale comenzilor acceptate de proces se realizează prin intermediul sistemului de interfaţă. In acest fel,calculatorul devine capabil pentru achiziţia,prelucrarea şi distribuţia datelor,avînd un control de tip ON-line asupra procesului.

Structura unui astfel de sistem de interfaţă,implementat pentru calculatorul TIM-S, este prezentată în fig.1.

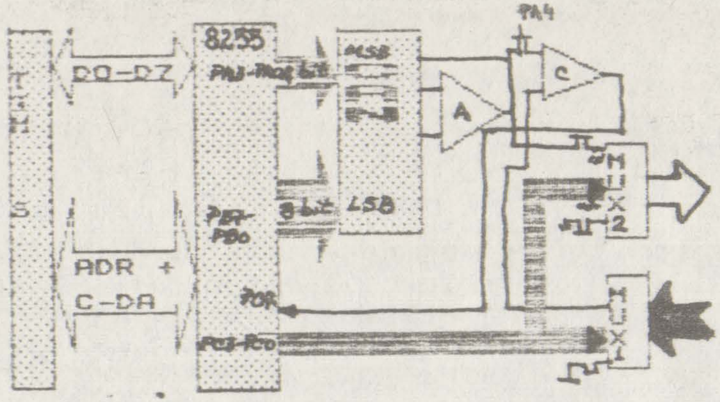


Fig.1Schema bloc a sistemului de interfaţă cu procesul.

Prin această configuraţie este posibilă achiziţia şi distribuţia a 16 mărimi analogice.Partea de achiziţie a datelor,în care intervin multiplexorul analogic MUX 1, convertorul numeric-analogic,CNA(cu amplificatorul de ieşire aferent, A),comparatorul C şi circuitul de interfaţă programabil 8255,este condusă prin software de către microcalculatorul TIM-S,fiind similar cu cea descrisă în (1).La aceasta structură,devine posibilă şi realizarea distribuţiei datelor,comandată de asemenea prin software,adăugînd un al doilea multiplexor analogic MUX 2. Acesta primeşte la intrare mărimile analogice în CNA şi le distribuie la 16 ieşiri de comandă ale procesului.Separarea celor două funcţii de achiziţie si distribuţie a datelor se realizează prin intermediul comutatoarelor K1 - K4, comandate printr-o linie disponibilă a portului A(de exemplu PA4),aferent circuitului 8255.

În fig.2 se prezintă o secvenţă posibila de conducere a unui proces, în care intervin fazele de achiziţie,prelucrare şi distribuţie a datelor.

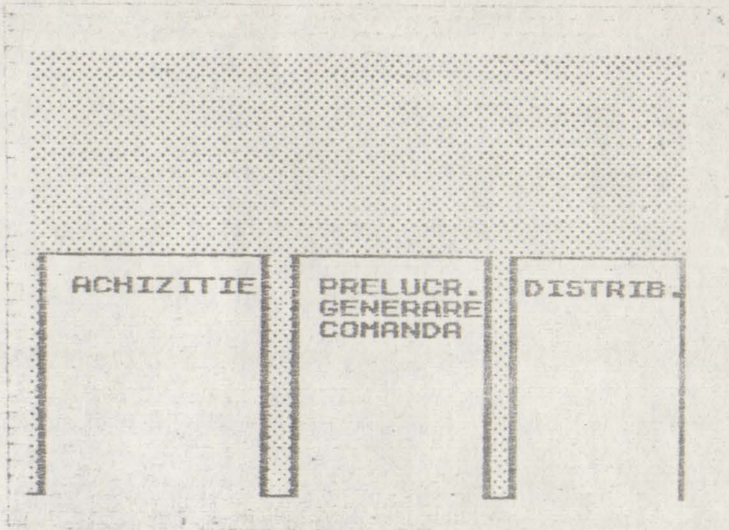


Fig.2 Secvenţa operaţiilor de conducere a unui proces.

În continuare se prezintă un program de achiziţie şi prelucrare de date pentru măsuratori de laborator,denumit DATA LAB.

Programul este divizat într-o suită de subprograme, după cum urmează:

- LD - MANUAL : manual de utilizare
- RULARE : program în cod maşină pentru antrenarea unei porţiuni a ecranului într-o mişcare comandată de programul principal (Autor:Stud.Dragomir Radu)

```
2 CLS
5 POKE 23613,0
10 FOR i=61000 TO 61033: READ
a: POKE i,a: NEXT i
20 DATA 6,0,62,0,7,7,7,79,22,2
0,30,5,197,205,170,34,183,245,24
1,203,30,35,245,29,32,248,241,19
3,4,21,32,234,201,0
25 POKE 23613,0
30 PAPER 2: BORDER 2: CLS : LO
AD ""
```

- CAN : programul de lucru cu interfaţa analog-digitală. Lucrează pe 12 biţi şi 16 canale. In cazul utilizării unei alte interfeţe analog-digitale, în acest loc se va introduce programul corespunzător, avînd grijă ca în programul principal să se modifice adresele de apelare (în subrutina 9000)

```
5 POKE 23613,0
10 CLEAR 62000
20 RESTORE 50
30 FOR a=62000 TO 62008
40 READ n: POKE a,n
50 DATA 62,136,211,255,62,1,21
1,223,201
60 NEXT a
70 FOR b=62020 TO 62054
80 READ m: POKE b,m
90 DATA 0,0,0,0,17,0,8,1,0,0,1
22,128,211,159,103,123,129,211,1
91,111,219,223,230,128,32,2,68,7
7,203,58,203,27,48,232,201
100 NEXT b
110 PAPER 0: BORDER 0: CLS
115 POKE 23613,0
120 LOAD ""
```

- CEAS : un ceas intern, afişează ora în colţul din dreapta sus a ecranului. Funcţionează continuu, cu excepţia momentelor de LOAD şi SAVE (Modificare a programului publicat în nr. anterior)

- DATA : programul de lucru propriu-zis.

2. Utilizarea produsului

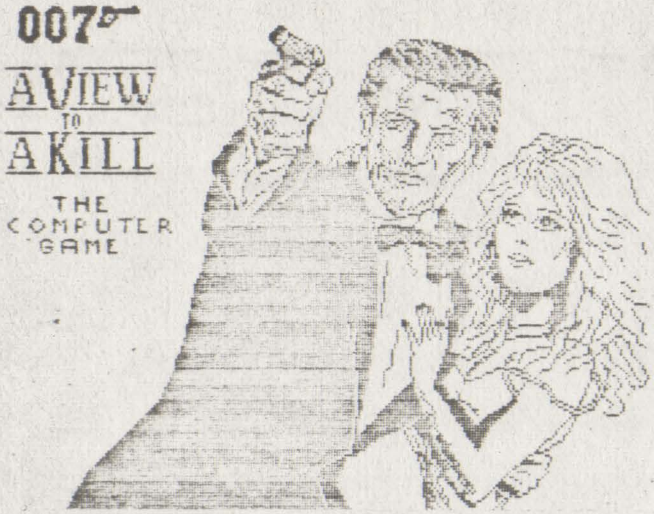
Incărcarea programului se face cu comanda LOAD. După ce se introduce manualul, se solicită oprirea casetofonului şi se întreabă dacă se doreşte vizionarea manualului. În caz că nu, se trece la încărcarea porţiunilor următoare.

Manualul este compus dintr-o suită de pagini afişate pe ecran, schimbarea lor făcîndu-se prin tastare.

Cînd încărcarea ajunge la CEAS, după ce acesta a fost introdus în memorie, se cere oprirea casetofonului, timp de circa 15 sec., după care se trece la încărcarea părţii finale.

Urmează iniţializarea sistemului de achiziţie date :

- ora de pornire: se introduce sub forma: 00 : MM : ØØ fixînd minutele (MM) cu aproximativ 1 min. înaintea orei reale, după care, cu cca 3 sec. înainte de coincidenţa celor două valori, se apasă tasta ENTER:




```
  12 INPUT "Introduceti ora de p
ornire      "'"Exemplu:  09.46.00
  ";a$: IF LEN a$<8 THEN PRINT "
GRESIT !  MAI INTRODUCETI": BEEP
 0.5,15: GO TO 12
  13 IF VAL a$( TO 2)>12 THEN PR
INT ;"ORA EXPRIMATA IN CIFRE 1-1
2": BEEP 1,10: GO TO 12
  14 POKE 63667,VAL a$(1)*16+VAL
 a$(2)
  16 POKE 63668,VAL a$(4)*16+VAL
 a$(5)
  20 POKE 63669,VAL a$(7)*16+VAL
 a$(8)
  25 RANDOMIZE USR 65040
 105 GO TO 1000
```

- numărul canalelor utilizate: max.16. Se va ţine cont că pentru canalele 1 - 4 programul este prevăzut cu o subrutină de reprezentare grafică a mărimii citite (în volţi 0-5 V), funcţie de timp. Pentru canalele 5 -16 se face doar raportarea pe ecran, ori imprimantă .

- intervalul de citire în minute: valoarea 0 are ca efect citirea continuă. Timpul de la pornirea sistemului se calculează pe baza valorii indicate pe ecran şi este afişat continuu.Programul lucrează în intervalul 0-24 ore.

După introducerea acestor valori,pe ecran apare o pagină de meniu:

1. Iniţializare sistem
2. Continuare măsură
3. Salvare date pe casetă
4. Ajustare ceas

Doar opţiunea 1 este operativă în această fază,iniţializarea se face pentru fiecare canal în parte, cerîndu-se următoarele date :

- factorul de conversie: dacă se citesc direct volţi,acesta se dă " 1 ". Dacă între tensiunea afişată şi mărimea reală există o dependenţă de tipul :

tensiune = constantă + mărime

se dă această constantă .

Pentru cazuri mai complexe,se intervine în subrutina 9000 şi se întroduce acolo transformarea, cu ajutorul instrucţiunii LET S (x) = f(x), unde x este numărul canalului.

- unitatea de măsură : sub forma unui şir alfanumeric de max. 2 caractere.

- există sau nu valoare la care se cere alarmarea : se răspunde prin " d " sau " n " . Dacă există, se cere introducerea ei.

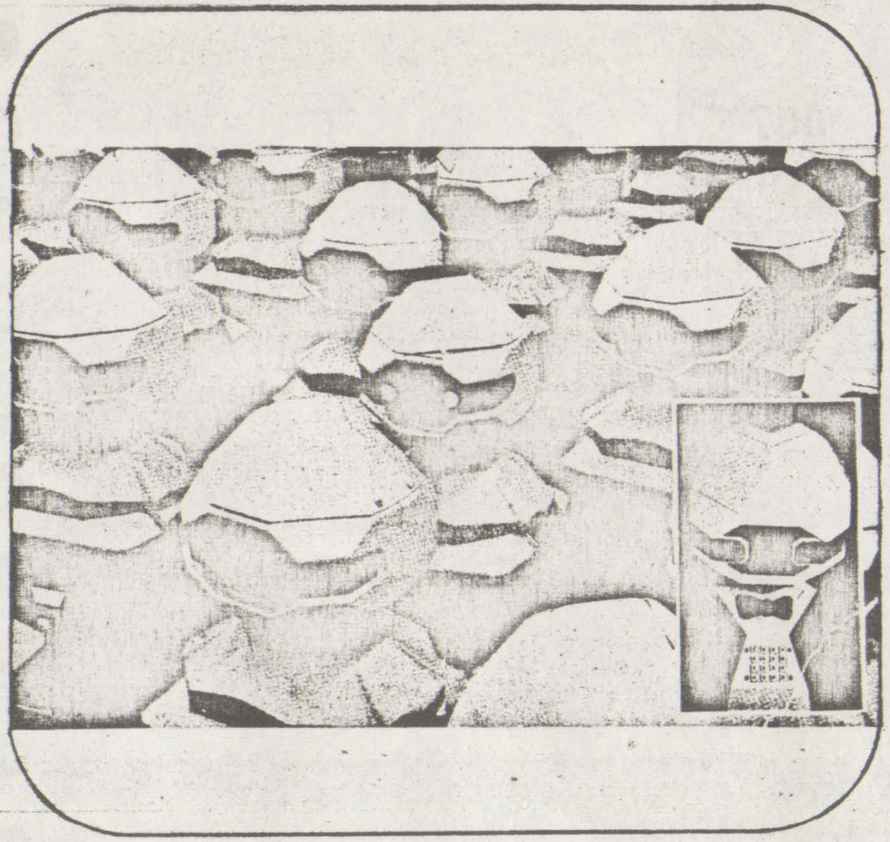
După introducerea acestor date, se cer opţiunile de tipărire.Programul poate memora un număr de 35o seturi de date,pentru 16 canale. O variantă perfecţionată permite memorarea de max.30.000 date,utilizînd porţiunea liberă a memoriei.Se reţin valorile USR citite în subrutina 9.000.

Pentru a profita de această facilitate, se răspunde " 1 " la întrebarea dacă se memorează datele şi se dă numărul de seturi de date memorat.Se va ţine cont aici de durata maximă a procesului măsurat şi de intervalul de citire. In cazul unei citiri continue, cînd s-a introdus intervalul " 0 ", citirea se face cu o periodicitate de 1 pînă la 15 sec.,funcţie de numărul de canale.

Cu acestea, sistemul este iniţializat şi gata de lucru.Pe ecran apare mesajul: " Pentru START tastaţi". În momentul în care se doreşte începerea achiziţiei datelor,se tastează şi apare ecranul de lucru, pe care se afişează:

- ora începerii, ceasul, timpul de cînd funcţionează sistemul,
- valoarea(în volţi) a mărimilor citite,
- 1 - 4 grafice cu variaţia mărimii măsurate pe canalele 1-4,în timp
- un meniu cu opţiunile: STOP şi MENIU principal,la care se ajunge tastînd " S ", ori " m ". STOP are ca efect oprirea temporară a sistemului,cu afişarea mesajului "PAUSE". Repornirea se face tastînd orice pe claviatură.

Dacă se memorează date, după parcurgerea numărului de citiri indicate,apare mesajul FULL(plin) şi opţiunile COPY şi MENIU,afişate alternativ,la care se ajunge tastînd " c " şi respectiv " m " , în cazul " c " , se obţine o copie-ecran, în prealabil trebuie pornită imprimanta.



Salvarea datelor pe casetă se face trecînd în meniul principal şi tastînd "3". Se salvează întîi un tablou din două elemente , conţinînd un vector n (nr.canale, nr.seturi) pentru iniţializarea unui program de prelucrare ulterioară,apoi matricea datelor W (nr.canal, valoare),după care programul revine la meniul principal.

ATENTIE : Programul poate fi oprit cu BREAK, numai dacă porţiunea DATA este încărcat separat,ori după începerea măsurătorii.

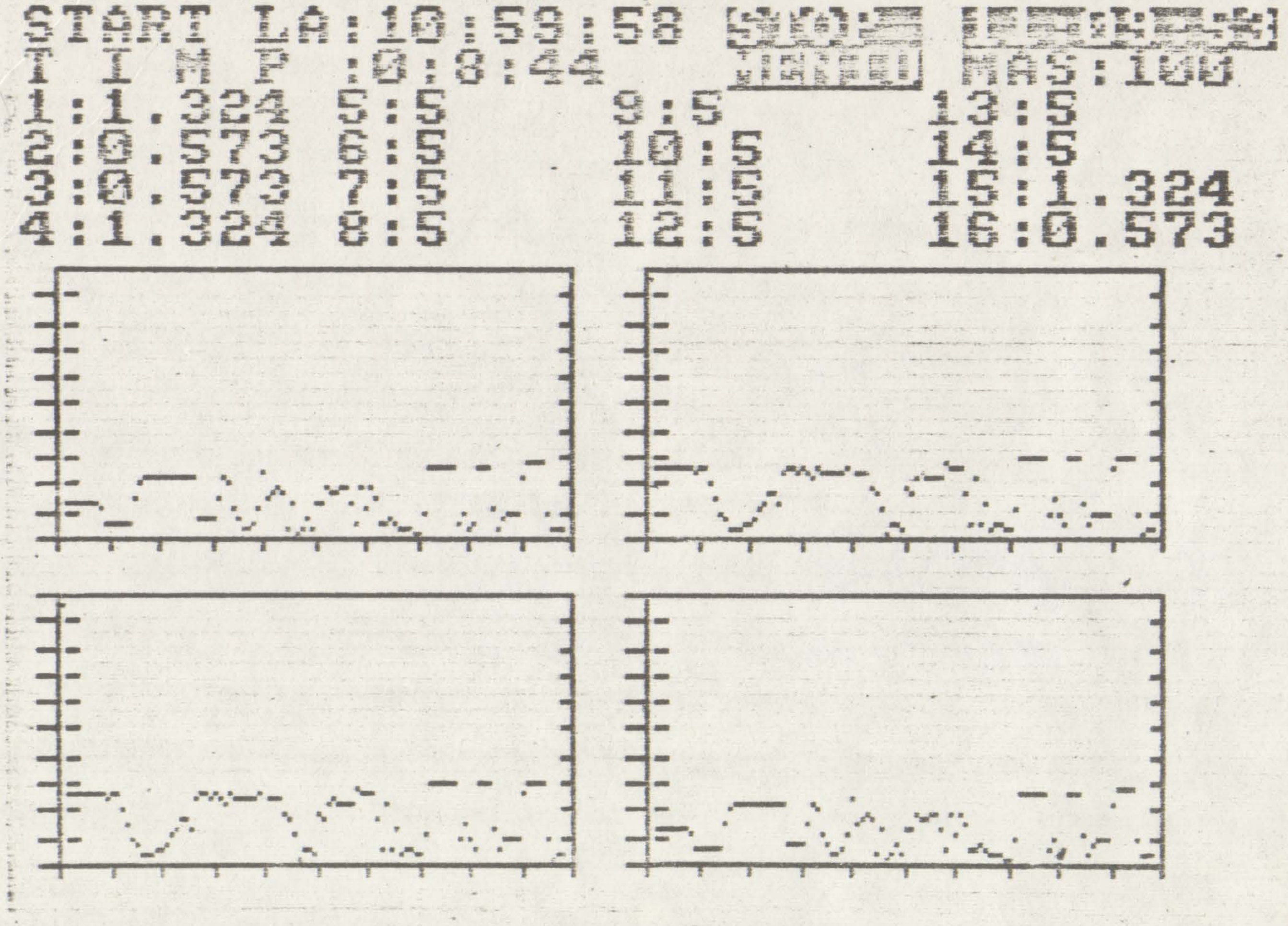


Fig.3. Ecranul de lucru al programului DATA LAB. Canalele 1-4 şi 15-16 sînt cuplate cu traductori fotoelectrici. Graficele reprezintă variaţia în timp a mărimilor măsurate pe canalele 1, 2, 3 şi 4.

3. Alcătuirea produsului

Pentru cei care doresc să aducă unele modificări în program, vom da în continuare o schemă sumară a alcătuirii acestuia :

```
  0 - 100 Iniţializare ceas
110 - 17o Secvenţa de început
170 - 280 Programul principal
SUB   300 Subrutină meniu de lucru
SUB  1000 Subrutină de iniţializare
SUB  2000 Subrutină citire ceas
```

```
2000 LET o$=SCREEN$ (0,24)+SCREE
N$ (0,25): LET m$=SCREEN$ (0,27)
+SCREEN$ (0,28): LET s$=SCREEN$
(0,30)+SCREEN$ (0,31)
2010 LET timp=3600*VAL o$+60*VAL
 m$+VAL s$
2020 RETURN
```

```
SUB  2030 Subrutină calcul timp
SUB  3000 Subrutină afişare
```

```
3200 POKE 61001,y: POKE 61003,x:
 POKE 61009,dy: POKE 61011,dx
3210 RANDOMIZE USR 61000
3220 RETURN
```

```
3310 PLOT 8,8+20*s(1)
3312 LET x=1: LET y=8: LET dy=10
0: LET dx=25: GO SUB 3200
```

```
SUB  4000 Subrutină alarmare la atingerea
          limitei maxime
SUB  5000 Subrutină lucru cu imprimanta
SUB  9000     "     lucru cu interfaţa
```

```
9000 FOR v=0 TO ncan-1
9005 LET i=v+1
9010 POKE 62005,v
9020 RANDOMIZE USR 62000
9030 GO SUB 9100
9040 NEXT v
9050 RETURN
9100 LET c=USR 62020
9110 LET s=c*5/4095
9115 LET s(i)=s
9117 IF salv=10 THEN LET w(i,set
)=s(i)
9120 RETURN
```

Utilizatorul poate interveni în program numai după ce se depăşeşte iniţializarea, altfel există pericolul distrugerii programului. În cazul în care afişarea se doreşte nu în volţi, ci direct în unitatea de măsură aleasă, se va interveni în zona de tipărire, după instrucţiunea 3360.

Repornirea programului, după oprire cu BREAK, se face cu GOTO 110 dacă se iniţializează timpul de lucru la " 0 " sau cu GOTO 170, dacă se păstrează valoarea iniţială.

4. Defecţiuni

Sînt posibile o serie de întreruperi, de exemplu dacă se introduce un şir alfanumeric în locul unei cifre. Repornirea, în cazuri de această natură, se poate face în două moduri:

- se apasă tasta CONTINUE (c)
- se dă GOTO x, unde x este o eticnetă imediat inferioară celei enunţate în mesajul de eroare.

În cursul exploatării programului nu s-au observat apariţia altor erori.

5. Necesităţi hard

Pentru exploatarea eficientă a programului sînt necesare următoarele componente:

- calculator TIM-S, ori compatibil
- casetofon
- monitor (alb-negru sau color)
- imprimantă(opţional,exploatarea eficientă a programului cere existenţa unei imprimante, însă în lipsă se poate stoca ecranul astfel: se întrerupe programul cu BREAK şi se comandă:SAVE "nume" SCREEN
- convertor analog-digital.(Dacă acesta nu este compatibil cu programul CAN, programul său de operare se va încărca în locul secţiunii CAN).

BIBLIOGRAFIE

(1)M.Crişan, M.Vlăduţiu: Sistem de achiziţie de date interfaţat cu microcalculatorul TIM-IMF, 1987.

VALLO LADISLAU

Calculatorul personal Commodore 4/+

Este construit pe o unitate centrală de 8 biţi (microprocesorul de tip 7501, succesorul lui 6502).

Pe lîngă Software rezident uzual, ca spre exemplu monitorul pentru scrierea programelor în cod maşină, intrepretorul BASIC V3.5 etc, acest calculator mai are implementate alte 4 programe (de unde îi vine numele Plus/4), printre care :

A) - Program pentru prelucrarea şi editarea documentelor (cu o amploare de 99 linii şi 77 caractere).

Ecranul TV putînd afişa simultan pînă la 22 linii a 37 caractere, devine în acest caz cursor (fereastră) care, printr-o tehnică specială de scrolling, poate fi deplasată în orice parte a documentului.

Programul permite ştergerea anumitor pasaje din text sau adăugarea unor texte suplimentare, salvarea documentului pe dischetă, de unde oricînd poate fi rechemat, precum şi formatarea la editare pe imprimantă a conţinutului documentului (modificarea bordajelor, centrarea unor anumite aliniate etc). În acest sens, s-au conceput instrucţiuni simple şi eficace.

B) - Program pentru calcule tabelare, bazat pe o variantă de calcul matriceal. Similar ca în cazul programului anterior, datele acestui calcul tabelar pot fi salvate pe dischetă şi la dorinţă tipărite.

Calculele pot fi efectuate pentru un bloc de 17 coloane de max. 11 caractere (la nevoie, prin concatenare, coloanele pot fi extinse la 37 caractere) şi 50 linii (deci, în total, 850 cîmpuri).

C) - Program pentru stocarea şi prelucrarea datelor de forma unei bănci de date cu pînă la 999 seturi de date, de maximum 17 cîmpuri (coloane) a cîte 38 caractere.

Calculatorul poate sorta datele introduse simultan după 3 cîmpuri. De exemplu, o listă de adrese poate fi sortată după nume, localitatea domiciliatoare, codul poştal etc. Banca de date poate fi păstrată pe dischete şi folosită la nevoie. Pentru tipărirea datelor din banca de date se apelează la programul de prelucrare a textelor anterior menţionat.

Prin cele menţionate mai sus, posibilităţile acestui computer nu sînt epuizate. Oricînd, disponibilul de date peste 60 KB-RAM facilitează efectuarea de calcule matematice, reprezentări grafice sau chiar generarea de sunete.

Pe lîngă caracterele alfa numerice uzuale, COMMODORE Plus/4 dispune de 62 simboluri grafice.

Intrarea în modul grafic se face cu instrucţiunea GRAPHIC, avînd formatul :

GRAPHIC(numărul modului grafic),(ştergerea ecranului DA,NU)

Calculatorul dispune de 3 moduri grafice de bază (grafica de text, grafica de înaltă rezoluţie - Hi Res - şi grafica policromă), existînd şi posibilitatea de combinare a acestora.

De asemenea, există posibilitatea scrierii unor texte în partea rezervată pentru grafică (instrucţiunea CHAR).

Principalele instrucţiuni de grafică sînt DRAW, BOX, CIRCLE, PAINT. De menţionat că în cazul graficii de înaltă rezoluţie ecranul TV, din punct de vedere al calculatorului, este împărţit în 40x8=320 puncte pe orizontală şi 25x8=200 puncte pe verticală, deci în total 64.000 pixeli.

Calculatorul are la dispoziţie 16 culori distincte, fiecare putînd fi redată în 8 nuanţe diferite (în total 121 variante).

Pentru generarea de sunete, acest calculator este echipat cu 2 generatoare de sunete: primul pentru reproducerea vocii 1, iar al doilea pentru reproducerea vocii 2 şi a zgomotelor (fîşîiturilor).

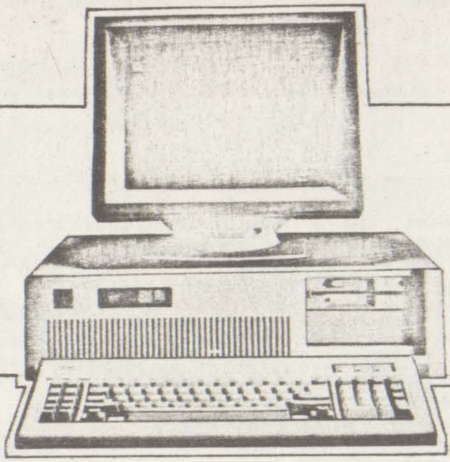
Sunetele pot fi reglate în 8 intensităţi diferite prin instrucţiunea VOL. Sintaxa instrucţiunii de sunet este de forma:

SOUND(vocea nr),(înălţimea de ton),(durata tonului)

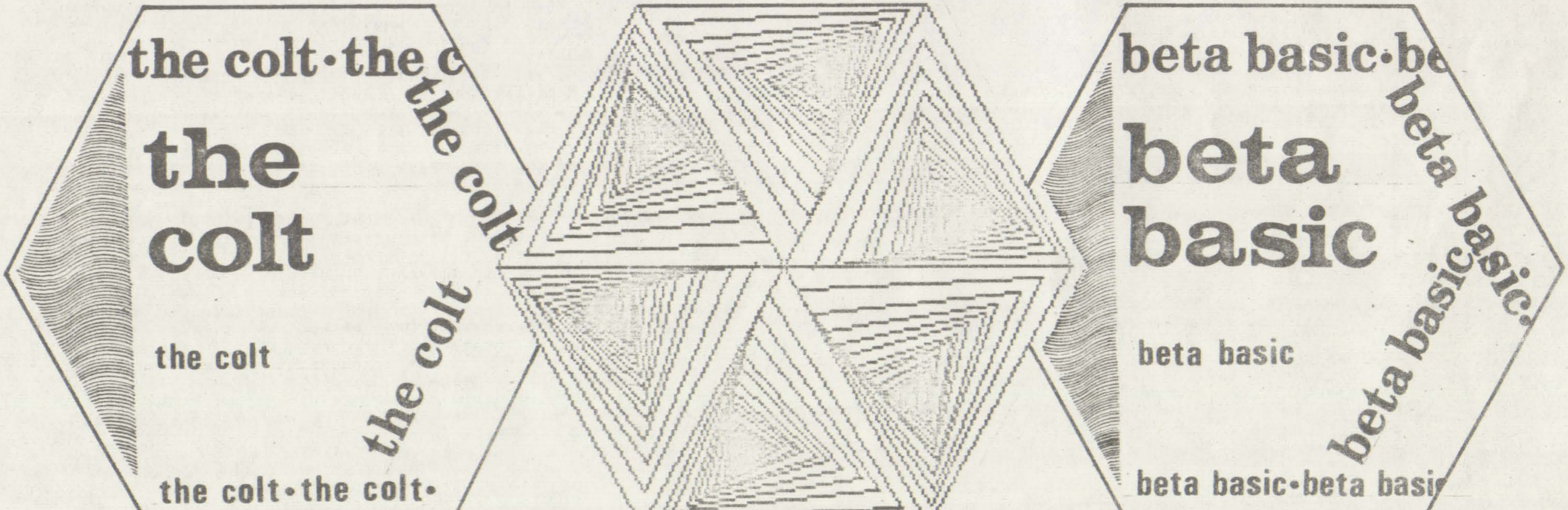
Ca periferic, calculatorul dispune de o unitate de disc flexibil de 5¼ ţoli de tipul 1551, cu o capacitate de formatare de 174848 Bytes/dischetă (683 blocuri a 256 Bytes).

Deosebirea acestui floppy de cel folosit anterior de firmă (1541) constă în conectarea lui la calculator (în acest caz, legarea se face prin intermediul unui adaptor la conectorul de interfaţă RS232 al calculatorului, faţă de sistemul cu IEC Bus serial, folosit anterior).

Astfel se reuşeşte transmiterea de 3 ori mai rapidă între calculator şi floppy-disc. Din păcate, acest floppy este compatibil numai cu calculatoarele Plus/4, C16 şi C116. De asemenea, software pentru floppy-disc 1541 nu poate fi folosit la 1551 fără o prealabilă adaptare.



MANUALE DE UTILIZARE



THE COLT

COLT ZX BASIC Compiler

Cuprins:

1. Introducere

COLT este un produs nou care transforma programele BASIC ale Spectrum-ului in cod masina pur accelerindu-le de 2 la 800 ori.

COLT este simplu de utilizat si suficient de scurt pentru a permite compilarea majoritatii programelor. In continuare se urmareste procesul de compilare pentru a vedea cit este de comod un program ca si COLT-ul.

2. Procesul de compilare

Cind scrii programul in BASIC pe Spectrum, ai avantajul ca poti "conversa" cu calculatorul, "interactionind" in program. Aceasta inseamna ca poti rula programul si corecta orice eroare in acest timp. Aceasta se poate realiza pentru ca sistemul BASIC folosit de Spectrum este cunoscut ca fiind "interpretor". Fiecare linie pe care o scrii este interpretata si daca are sens atunci este introdusa de program in memorie. "Interpretorul de sintaxa" care tipareste semnul '?' ori de cite ori esti pe cale de a introduce o linie gresita, poate detecta numai ceea ce se numeste "eroare de sintaxa". O eroare de sintaxa este o eroare de program care nu poate alcatui o parte din programul BASIC pentru ca nu corespunde regulilor de sintaxa ale limbajului.

Interpretorul de sintaxa nu poate detecta erori de algoritm, care sint erori de programare, asa ca desi BASIC-ul este valabil , nu poate rezolva ceea ce tu astepti sa faca. De ex. poti avea o linie in care 10 este alaturat unei variabile, pe cind in realitate, tu ai alaturat 5. Acestea sint in general erorile cele mai greu de gasit.

Odata ce ai programul BASIC bun, in memoria Spectrum-ului, poti executa programul apasind tasta 'RUN' si 'ENTER'. Pot apare erori, daca uiti afara linii sau faci calcule gresite. Programul se va opri in acest caz si-ti va spune in ce linie a aparut eroarea . De aceea sistemul este cunoscut sub numele de "interpretor" pentru ca iti permite sa-ti corectezi erorile pe masura ce apar, pentru a putea rula in continuare programul corect.

Cind apesi 'RUN' pentru un program, interpretorul gaseste prima linie din program, care este stocat in computer si executa ceea ce i se cere. Poate sa continue cu alta linie, poate chema subroutine sau poate doar sa tipareasca ceva pe ecran si sa termine. Nu conteaza ceea ce face, dar conteaza cum face ceea ce face. Programele de interpretare sint mai lente decit posibilitatile reale ale computerului pentru ca fiecare linie este examinata de acest program de interpretare, ori de cite ori o intilneste. Bucla FOR-NEXT nu este convertita automat intr-o bucla in cod-masina care se termina dupa un anumit numar de iteratii, dar este examinata cind o linie care contine NEXT este intilnita si interpretorul vede daca capatul a fost atins. Interpretorul BASIC din Spectrum se afla in ROM si nu poate fi usor remutat. COLT suplineste acest interpretor prin compilator.

Compilatorul este un program care transforma programele scrise intr-un limbaj particular in cod-masina, asfel incit computerul poate rula aceste programe direct, mai bine decit trecindu-le prin interpretor. In cazul compilatorului COLT, limbajul convertit este BASIC-Spectrum. Rezultatul final este un

program in cod-masina cu mult mai scurt, care face exact ceea ce face si programul din BASIC dar mult mai repede. Precum se stie, in cod-masina nu se poate rula un program apasind tasta RUN, pentru ca aceasta este o comanda de interpretor. De aceea, este nevoie de utilizarea unei adrese data de compilator. Putem chema programele din cod-masina cu declaratia RANDOMIZE USR <adresa>, unde <adresa> este adresa pe care o dorim.

Sint doar foarte putine caracteristici BASIC pe care COLT-ul nu le poate manevra, dar acestea se vor studia mai tirziu. La inceput, sa vedem cum lucreaza intregul sistem.

3. Pornirea cu COLT

Compilatorul Colt contine si un scurt program BASIC menit sa incarce compilatorul in memorie si sa-l initializeze.

```
LOAD "" <ENTER>
```

Intii, se incarca programul BASIC care are auto-RUN. Primul lucru pe care il face este incarcarea compilatorului in computer. cind intregul proces este terminat, vei fi intrebat daca vrei sa copiezi compilatorul pe alta caseta sau pe micro-drive. Apoi vei fi intrebat daca vrei sa schimbi adresa de pornire a programului compilat de la 40000. Cit timp nu se compileaza programe lungi este bine sa nu se schimbe aceasta adresa. Ecranul se sterge si apare un ceas in coltul drept al ecranului, care este o parte din programul Executor pe care-l studiem mai tirziu. Acum tipareste urmatorul program:

```
 10 LET h$="0123456789ABCDEF"
 20 FOR a=0 TO 255
 30 LET d$="  ": REM doua spatii intre siruri
 40 GO SUB 100
 50 POKE 23692,255
 60 PRINT a;" = ";d$
 70 NEXT a
 80 STOP
100 LET x=INT (a/16): LET y=a-16*x
110 LET d$(1)=h$(x+1)
120 LET d$(2)=h$(y+1)
130 RETURN
```

Cind vei rula acest program, vei vedea numerele de la 0 la 255 tiparindu-se cu echivalentele lor hexazecimale. Daca cronometrezi acest program, vei vedea ca dureaza doar 30 de sec.

Pentru a compila programul, tipareste RANDOMIZE USR 60000 apoi apasa ENTER. Dupa foarte scurt timp vei vedea aceasta informatie pe ecran:

```
                    00:00:00:0
HISOFT COLT Integer Compiler 1.0
c 1985 THRELFALL and HODGSON

Compiling Line 130

Compiled O.K.

   Begin  End    Vars   Lines
   40000  40384  51529  52003

   Names  Nmtop  Ctop   Nvars
   52059  52191  52460  80

To run RANDOMIZE USR 40000
```

(unele din aceste numere si mesaje pot sa difere)

Toate aceste numere folosesc programatorilor avansati, iar noi putem sa le ignoram. Tot ce dorim sa stim este ca mesajul 'Compied OK' a aparut. Daca o linie nu poate fi compilata (pentru ca nu are sens ori pentru ca este una din putinele cazuri pe care COLT nu le poate manevra) atunci linia dubioasa este reprodusa pe ecran alaturi de '?'. Programul de mai sus, daca este tiparit corect, nu va produce erori.

Cind toate de mai sus apar pe ecran, rularea programului compilat se face prin : RANDOMIZE USR 40000 (apoi tasta ENTER). Veti vedea diferenta imediat! Desi exemplul dat este executat de doua ori mai rapid decit BASIC-ul, alte programe pot atinge viteze de 70 de ori mai mari. Cind programul compilat se termina , poti tipari LIST si sa afli daca programul original se afla tot acolo, gata pentru a fi compilat din nou.

4. COLT in detaliu

COLT este in totalitate compatibil cu microdrive si interfata 1. El va compila aproape tot codul ZX BASIC inclusiv sirurile, toate comenzile de la microdrive, comenzile de pe taste, comenzile RS232, comenzi asociate cu Executorul.

COLT este un compilator pentru numere intregi in BASIC, ceea ce inseamna ca poate opera doar cu numere intregi intre -32768 si 32767. Aceasta s-a realizat a. i. viteza maxima care se poate obtine, sa faca COLT-ul ideal pentru scrierea jocurilor. COLT a fost conceput sa nu foloseasca virgula aritmetica flotanta, astfel cresterea vitezei fata de interpretorul Spectrum ar fi neglijabila.

Colt include multe caracteristici speciale in mentinerea compatibilitatii cu BASIC-Spectrum. Nu exista comenzi speciale care sa nu poata fi testate in circumstante normale.

Urmeaza o lista cu citeva dintre caracteristicile COLT-ului:

- Manevrarea totala a sirurilor.
- Pina la 26 de matrici unidimensionale si pina la 26 de matrici siruri.
- Compatibil in totalitate cu microdrive si interfata 1.
- Toate comenzile microdrive cu exceptia DATA, pot fi compilate.
- Toate numele de variabile.
- Acces la variabilele BASIC.
- BASIC-ul are acces la variabilele COLT.
- INPUT cu intreaga linie editata.
- Estimeaza GO TO <expresie>.
- Se pot folosi VAL si VAL$.
- Tasta BREAK se poate folosi in permanenta.
- Compileaza pina la 32 Ko din BASIC.
- Compilare foarte rapida (max. 2 sec. pentru 1 Ko BASIC) cu raportarea erorilor gasite.

5. Incarcarea compilatorului

Se face apasind comanda LOAD "". Se va incarca astfel un program BASIC care da unele informatii despre compilator. Apoi ti se ofera posibilitatea sa copiezi COLT-ul pe caseta sau pe microdrive. Aceasta iti permite sa tratezi copia ce ai facut-o ca si MASTER BACKUP. Ti se ofera sansa sa schimbi adresa la care compilatorul plaseaza codul de compilare. Aceasta este RAMTOP si pentru compilare poate fi intre 26000 si 59000. Daca folosesti Executorul, limita superioara este 52000.

6. Declaratii in COLT

Numele de variabile pot fi de orice lungime dar pot contine numai caracterele: de la A la Z, de la a la z si de la 0 la 9 (numerele nu pot fi primul caracter din numele variabilelor). Ca si la BASIC, literele mari si literele mici sint tratate fara nici o diferenta. In numele variabilelor nu trebuie sa apara vreun cod de culoare, altfel compilatorul obiecteaza. In lista urmatoare :

'e' reprezinta o expresie arbitrara
'N' reprezinta un numar inreg si pozitiv
'a$' reprezinta orice expresie de variabile siruri sau siruri taiate (ca in 'x$ (1 TO 4)')
'a' reprezinta orice variabila simpla sau de bucla
'a(e)' rep orice matrice monodimensionala sau elemente de matrice

Codurile de culori si alte coduri de control sa nu apara in mijlocul liniei, decit intre ghilimele.

Cind 'x', 'y' si 'z' sint date ca argumente ale unei functii atunci virgula flotanta se poate evalua. Expresia 'n/m' poate inlocui 'x', 'y' sau 'z' si 'n/m' este calculat pentru a da un rezultat cu virgula flotanta. 'n' si 'm' pot fi variabile intregi sau expresii intregi intre paranteze, de ex:

'BEEP (1+2)/(3+4),5' va suna 3/7 dintr-o secunda
'BEEP 1/3,5' va suna 1/3 dintr-o secunda.

Semnul de divizare este unicul operator care este permis in afara parantezelor si acolo poata fi numai per argument (x,y,z). Expresiile siruri trebuie sa nu contina paranteze - de ex : a$=(b$+c$) - pentru ca sint inutile si pot incurca compilatorul. In declaratia PRINT comparatiile de siruri trebuie sa fie intre paranteze - de ex: 'PRINT ("x"<"y") este mai bine decit 'PRINT "x"<"y"'.

Numai daca nu se opresc din alte cauze, functiile de mai jos functioneaza exact ca si in BASIC. Acest tabel va fi examinat impreuna cu manualul de la Spectrum.

AND	- 'AND' boolean. Nu poate fi folosit pentru a amesteca siruri cu numere de ex: 'b$=a$ AND 2' nu va fi compilat.
ABS e	
AT e,e	
ATTR (e,e)	
BEEP x,y	- Vezi comentariile despre virgula flotanta de mai sus.
BIN e	
BORDER e	
BRIGHT e	
CIRCLE x,y,z	
CHR$ e	
CLS	- CLS# la interfata 1.
CAT	- Toate formele sint suportate.
CLEAR	- Numai variabilele de cod compilat sint afectate. CLEAR# pentru interfata 1. Forma 'CLEAR 12345' nu este suportata.
CLOSE	- Toate formele sint suportate.
CODE a$	
CONTINUE	- Nu foloseste in program dar poate fi folosit ca sa marcheze capatul codului ce este compilat. Trebuie sa fie prima sub-declaratie intr-o linie de multiple declaratii.
COPY	
DATA list	- 'list' trebuie sa fie o lista de nr intregi sau siruri intre ghilimele si nu au voie sa contina expresii.
DEF FN	- Functiile "user-defined" nu sint suportate.
DIM a(v)	- Numai matrici unidimensionale.
DIM a$(v)	- Despre matrici si siruri se va discuta mai tirziu. Toate matricile, sirurile si variabilele sint sterse cind se reintroduce programul COLT.
DRAW x,y	
DRAW x,y,z	
ERASE	
FLASH e	
FN	- Functile "user-defined" nu sint suportate.

FORMAT	- Toate formele sint suportate.
FOR a=u TO v STEP w	- 'STEP' este optional. Diferenta intre u si v sa nu depaseasca 32767.
Go SUB n	- 'n' este un nr intreg si pozitiv. Este o forma rapida a GO SUB.
GO SUB e	- Expresia 'e' sa nu inceapa cu un caracter numeric. Aceasta optiune este lenta si se va folosi doar exceptional.
GO TO n	- Ca si la GO SUB n,
GO TO e	- Ca si la GO SUB e.
IF e THEN	
IN e	
INK e	
INKEY$	
INKEY$# e	- Pentru interfata 1.
INPUT a INPUT a(v) INPUT a$	- Se editeaza exact ca in BASIC. Operatiile de colorare si toate cele trei forme de INPUT pot fi interschimbabile. Sufixul #e permite citirea din RS232 sau din cimpurile de la microdrive.
INT e	- Ajuta testarea in diverse imprejurari.
INVERSE e	
LEN a$	- 'a$' nu poate fi taiat.
LET a=e	
LET a$=a$	
LINE	
LOAD	- Nu se poate folosi forma LOAD "" DATA. Vezi cap. '18. Comenzi de incarcare'.
LPRINT	- Cu sau fara #e.
MERGE	
MOVE	- Toate formele sint suportate.
NEW	
NEXT a	- Vezi 'FOR'.
OPEN	- Toate formele sint suportate.
OR	
OUT e,e	
OVER e	
PAPER e	
PEEK e	
PLOT e,e	
POKE e,e	
POINT (e,e)	
PRINT	- Cu sau fara #e.
RANDOMIZE	
RANDOMIZE e	
READ a	- Citeste o lista de nr intregi de la declaratia DATA.
READ a$	- Idem, doar ca DATA sint siruri.
REM	- Se foloseste pentru tasta BREAK.
RESTORE e	- Returneaza datele la linia e. Linia e trebuie sa aibe o declaratie DATA.
RESTORE	- Returneaza datele pentru DATA de la inceputul programului.
RETURN	
RND	- Obtine un nr intreg aleator intre 0 si 32767 (nu este la fel ca si in BASIC). Pentru a obtine acelasi efect ca in BASIC foloseste USR 59997.
SAVE	- Nu se poate forma SAVE "" DATA.
SCREEN$ (e,e)	
SGN e	
STOP	
STR$ e	- Vezi comentarile la siruri si cap 'Virgula flotanta ?' de mai jos.
TAB e	
TO	- Toate formele a$(m TO n), a$(TO n) si a$(m TO) sint permise.
USR e	
USR "siruri"	
VAL a$	- Daca a$ contine o referire la o variabila atunci variabila BASIC cu acest nume este folosita. Vezi capitolul 'Aducerea variabilelor din BASIC'.
VAL$ a$	- Vezi 'VAL a$'.
VERIFY	- Nu se poate forma VERIFY "nume" DATA.

Toate comenzile Executorului pot fi compilate. Atentie totusi '* fx GO TO e' intodeauna se intoarce in linia 'e' din BASIC, nu la linia 'e' din codul compilat.

7. Siruri si matrici

Matricile si sirurile sint stocate in spatiul variabilelor BASIC (in spatiul marcat de variabila de sistem 'VARS' (adresa 23627 si 8)) si sint la inceputul acestui spatiu. Spatiul cerut pentru matrici si siruri este alocat dinamic ca o zona necesara in timpul de rulare, ca si in SINCLAIR BASIC. Este doar o mica diferenta: daca un sir sau o matrice devin mai mici, spatiul alocat lor nu descreste. Acest mod de lucru relizat in cod este de cinci ori mai rapid. In general manevrarea sirurilor se realizeaza de 20-25 de ori mai rapid ca in BASIC. Faptul ca spatiul nu este inapoiat, nu este observat de utilizator si ca cercetare 'LEN a$' va da rezultatul asteptat. Efectul este ca tu poti rula afara din spatiul alocat mai devreme ca de obicei, acest proces de manevrare a sirurilor rezultat in cod fiind mai rapid decit orice alt compilator BASIC.

Cu matricile se lucreaza in mod analog dar dureaza mai mult daca ele sint relocate si vechiul spatiu refolosit.

Spatiul folosit intre 'STKEND' si 'RAMTOP' (din sistemul de variabile BASIC) in timpul rularii este spatiul folosit temporar in formarea expresiilor siruri si este constituit pina la 256 octeti peste 'STKEND' (sfirsitul BASIC-ului normal). Pentru ca acest interval este de 256 octeti, nici un sir sa nu varieze in lungime peste 255 de octeti (caractere) intr-un singur drum.

8. Mesaje de eroare in timpul compilarii

Daca compilarea se opreste la o linie ce contine erori comanda EDIT o aduce in partea de jos a ecranului pentru a fi reeditata.

Declaratii de eroare in timpul compilarii:

1 Next without For
O declaratie 'NEXT' foloseste o variabila care nu a fost vazuta in declaratia 'FOR'.

2 Variable not found
Numarul maxim de variabile (255) a fost depasit - deci trebuie folosite mai putine variabile.

4 Out of memory
Lungimea tabloului numelor de variabile nu este destul de mare (schimba locatia 59991 si 2 - vezi cap 21).

8 End of file
Nu exista cod (program) de compilat.

C Nonsense in Basic
Compilatorul nu poate compila declaratii particulare, vezi capitolul 6.

G No room for line
Spatiul alocat codului obiect este prea mic; incearca o scadere a RAMTOP-ului cu 'CLEAR n' cu n<40000 .

M RAMTOP no good
Nu este loc pentru tablouri. Trebuie sa fie incapator spatiul pentru numele, liniile si variabilele de deasupra RAMTOP cind incepe compilarea, chiar cind se compileaza 32 Ko din cod.

N Statement not found
GO TO sau GO SUB pentru un numar de linie mai mare de 32767. Aceasta se poate intimpla si in timpul rularii cind avem GO TO (expresie) sau GO SUB (expresie).

9. Mesaje de eroare in timpul rularii

3 Subscript wrong
Atentie, numai matricile foarte simple pot mari viteza codului (se efectueaza doar verificari rudimentare asupra tablourilor, pentru a creste viteza).

6 Number too big
In general impartiri cu zero.

B Integer out of range
Un mumar mai mare decit 32767 sau mai mic de -32768 .

D and L BREAK into program
Intreruperea programului cu tasta BREAK .

E Out of data
O declaratie READ a fost incarcata dupa ce toate declaratiile DATA s-au terminat.

H STOP in INPUT
Instructia 'STOP' a fost introdusa in interiorul unei declaratii INPUT.

P FN without DEF
O rutina din Executor a fost chemata, dar Executorul nu exista sau nu este activat.

10. Salvarea codului compilat

Daca vrei sa salvezi codul compilat trebuie sa salvezi atit codul cit si compilatorul. Este mai usor sa salvezi de la RAMTOP (in general 40000) la 65535 asa cum se salveaza orice definire grafica. 'Elk' din Executor va face aceasta automat, altfel comanda va fi:

```
LET rt=USR 59200: SAVE "mane" CODE rt, 65535-rt: LET rt=59200
```

Nu apasa BREAK in timpul salvarii fiindca compilatorul se poate distruge.
Pentru a micsora totalul codului ce trebuie salvat incearca sa faci 'VARS' numai cu putin mai mare decit 'END' (vezi cap 21). Aceasta se poate face cu:

```
CLEAR (PEEK 23730+256*PEEK 23731)+(VARS-END-5)
```

- unde 'VARS' si 'END' sint citite de pe ecran la sfirsitul compilarii. Codul trebuie recompilat si noua adresa de intrare notata.

11. Reincarcarea codului compilat

Este foarte important ca inainte de reincarcarea codului intr-o masina "curata" sa faci 'CLEAR n' unde 'n' este RAMTOP-ul de la care codul tau a fost compilat (in gen. 40000). Nefacind acest lucru codul-masina va fi spart. Daca codul compilat pe care il reincarci contine comenzi de Executor, atunci Executorul trebuie repornit cu 'RANDOMIZE USR 55020'. Acest lucru nu este necesar pentru functiile "sprite" si de fereastra.

12. Facilitati de urmarire

O trasatura foarte folositoare a COLT-uli este aceea ca il poti folosi cu sau fara posibilitatea de intrerupere prin

'BREAK' si in plus, in asociatie cu Executorul, se poate urmari si vedea numarul liniei ce urmeaza a fi executata.
Acestea se obtin cu trei declaratii 'REM', fiecare de forma 'REM #n':

REM #0
BREAK este imposibil, intreruperea se face la 'scrool ?', INPUT si in timpul comenzilor de lucru cu casetofonul sau microdrive. Astfel rezulta un cod mai rapid mai sigur si mai scurt. Programul poate fi intrerupt cu Executorul prin 'E!x'.

REM #1
BREAK este posibil. Aceasta determina un cod mai lung decit REM #0 dar tasta BREAK lucreaza normal.

REM #2
Cind se foloseste in asociatie cu comanda Executorului '* fx L', numarul liniei executate este actualizat de 50 de ori pe secunda. Aceasta este bine de folosit la debutul programului. In caz de eroare in timpul rularii, se afiseaza numarul liniei ce contine erori (nu se afiseaza si numarul declaratiei din linie).

13. Compilarea pina la 32 Ko

In general este posibila compilarea pina la 16 Ko (chiar mai putin daca se foloseste si Executorul). Aceasta deoarece memoria trebuie sa tina atit sursa codului (programul BASIC) cit si codul compilat care are aproape aceiasi marime cu sursa BASIC, plus compilatorul.
Lucrind cu casetofonul lucrurile se schimba si programul in BASIC poate fi stocat pe caseta si reincarcat si modificat daca rezultatele nu sint cele asteptate.
Pentru compilarea a 30 Ko sau 34 Ko se procedeaza in felul urmator:
1) Salveaza programul in BASIC pe caseta fiindca in timpul compilarii el se sterge.
2) Cu 'CLEAR n' se pune RAMTOP astfel incit sa fie un spatiu suficient intre 'VARS' si 'Ctop' (vezi cap 21). Pentru un program de 30 Ko acest spatiu trebuie sa fie de aproximativ 5 Ko. Deci 'CLEAR 55000'.
3) Modifica adresele din COLT de la 59987 si 8 cu valoarea RAMTOP-uli de la care vrei sa intre codul compilat. De ex.: pentru RAMTOP 32767, (32767=127*256+255) se face

```
POKE 59987,255: POKE 59988,127
```

pentru RAMTOP 55000 (55000=214*256+216)

```
POKE 59987,216: POKE 59988,214
```

4) Compileaza. Daca compilarea reuseste atunci spatiul cu programul in BASIC se sterge. Daca apar erori in procesul compilarii atunci BASIC-ul trebuie sa ramina. Mesajele de eroare apar in modul normal descris anterior.
5) In timpul compilarii stiva masinii va fi mutata de la adresa aleasa de tine la 65535 si apoi la o noua valoare aleasa.
6) Inainte de reincarcarea oricarui program trebuie executat pasul (2), altfel nu va fi spatiu pentru BASIC.
7) Daca ai nevoie de Executor atunci se pot compila doar aproximativ 25 Ko BASIC si la pasul (2) se face 'CLEAR 48000'. Pentru a sterge Executorul ca sa fie mai mult spatiu (pentru compilarea a 32 Ko BASIC) se face: RANDOMIZE USR 59190 (ENTER)

14. Revenirea in BASIC

O parte din codul compilat va fi intors automat in BASIC cind va fi gata. STOP produce mesajul de "eroare" (9). Compilarea unui program se executa pina la o declaratie CONTINUE daca aceasta exista. Daca codul compialat a fost chemat cu o comanda nenumerotata atunci nici o declaratie dupa 'RANDOMIZE USR n' nu va mai fi executata. Daca codul compilat a fost chemat de o linie din program (aflata dupa declaratia CONTINUE), atunci el se intoarce la declaratia urmatoare. Daca dorim sa se intoarca la o anume linie si la o anumita declaratie din acea linie atunci trebuie sa facem POKE 'OLDPPC' (adresa 23662 si 3) cu numarul liniei si POKE 'OSPPC' (adresa 23664 si 5) cu numarul declaratiei din linia respectiva. Aceste POKE-uri trebuie sa se afle in codul compilat. Aceasta metoda poate fi folosita la inlantuirea programelor. Daca codul compilat contine o linie 'LOAD "project"', si codul compilat este chemat de prima declaratie de la linia 10, atunci compilatorul se va intoarce la urmatoarea declaratie din linia 10 (daca exista) sau la urmatoarea linie din "project". De aici rezulta ca tu poti sa inlantuiesti BASIC-ul cu codul compilat. Atentie, comanda LOAD aflata in codul compilat, distruge sirurile si matricile acestuia.

15. Aducerea variabilelor din BASIC

In multe situatii este necesar sa culegi variabile din BASIC pentru a le folosi in programul COLT. In acest scop daca argumentul functiei VAL contine o referinta la o variabila, foloseste mai bine o variabila BASIC in locul uneia COLT. Atunci vei scrie asa:

```
LET fred = VAL "fred"
```

- ceea ce transfera variabila BASIC 'fred' intr-una COLT 'fred'. La fel in cazul elementelor unei matrici :

```
LET b(10) =VAL "b(10)"
```

Pentru a aduce sirurile BASIC in COLT ,se foloseste functia 'VAL$' :

```
LET a$ = VAL$ "a$"
```

16. Trecerea variabilelor COLT inapoi in BASIC

Nu este la fel de simpla ca operatia de la cap.15, de aceea s-au prevazut functii speciale ajutatoare:

```
LET fred=USR 59227: PRINT "john"
```

- ceea ce va egala variabila Basic 'fred' cu variabila Colt 'john'. La fel pentru matrici si siruri:

```
LET point=USR 59227: PRINT "b(n)"
```

- face ca 'point' sa fie un numar intreg in PRIMUL membru al matricei 'b' din COLT; 'n' poate lua orice valoare in afara de numarul care apartine intotdeauna primului membru. Pentru a ajunge la primul element al matricei 'b', este suficienta o linie BASIC:

```
LET b(1)=PEEK point + 256*PEEK(point+1)
```

Pentru a ajunge la membrul 'n' se foloseste:

```
LET b(n)=PEEK (point+2*(n-1)) + 256*PEEK (point+2*(n-1)+1)
```

- factorul lui 2 exista deoarece sint 2 octeti pe un membru al matricei.

Pentru a avea un sir, se scrie:

```
LET point=USR 59227: PRINT "a$"
```

- unde 'point' indica primul membru al sirului COLT 'a$'(de ex a$(1)), 'point+1'-al doilea membru (a$(2)), etc.

17. Virgula flotanta ?

S-a precizat deja ca programul COLT foloseste doar numere intregi aritmetice, deoarece virgula flotanta are nevoie de un timp prea lung de compilare. Exceptie fac comenzile DRAW si BEEP.

Functiile VAL si STR$ permit accesul la virgula flotanta totala aritmetica. Un numar cu virgula flotanta se tipareste cu:

```
PRINT STR$ VAL "SQR 3"
```

Functia VAL se poate folosi pentru a face valabile functiile cu virgula flotanta cum sint SIN, COS si TAN. Folosirea functiei VAL in mod normal da un rezultat intreg :

```
LET a = VAL "100*SQR 3"    - a va rezulta 447 !
```

Totusi, 'STR$ VAL' poate face ca rezultatul sa fie in virgula flotanta in cadrul unui sir:

```
LET a$ = STR$ VAL "100/3"
```

- a$ va fi sirul "33.333333". Acest sir poate fi manipulat mai departe astfel:

```
LET b$ = a$ + "*2"
LET c$ = STR$ VAL b$
```

- sirul c$ este de doua ori valoarea numarului inclus in sirul a$, adica "66.666666". Acest proces poate continua la infinit in cadrul unui program.

Orice variabila folosita intre ghilimele, ca de ex. "a*b", se refera la variabilele BASIC 'a' si 'b' si nu la variabile COLT. Aceasta formă este folositoare, dar va rula cu aceasi viteza ca si in BASIC. Citeva exemple:

```
10 LET a = VAL "SIN (.523598775)*100+100*.01"
20 PRINT a
```

- rezultatul tiparit va fi 51 si cu interpretor sau cu compilator. Expresia nu contine nici o variabila. Ea se poate complica astfel:

```
10 LET b = 100
20 LET a = VAL ("SIN (.523598775)*100+.01*"+STR$ b)
30 PRINT b
```

- care foloseste variabila compilata 'b'. Totusi, daca linia 20 se schimba cu

```
20 LET a = VAL ("SIN (.523598775)*100+.01*b")
```

- atunci este folosita variabila BASIC 'b'.

In exemplul de mai sus rezultatele sint numere intregi, dar folosindu-se siruri putem obtine rezultate in virgula flotanta astfel:

```
10 LET a$ = "100/LN 12 + EXP 2"
20 LET b$ = "SIN 1 + COS 2 + TAN 3 + .01"
30 LET c$ = VAL (a$ + "2.34973 +" + b$)
40 PRINT c$
```

-astfel rezultatul este 300 atit in BASIC cit si in codul compilator.

Este incomod si greoi dar merge !

(exemplele au fost copiate riguros din manualul original de utilizare al compilatorului - nota traducatorului)

18. Comenzi de incarcare

Singura comanda de incarcare care poate fi compilata fara restrictii este 'LOAD "" CODE'. Comanda pentru incarcat BASIC, 'LOAD "" ' va lucra bine dar va sterge toate matricile si sirurile COLT din zona 'VARS' si din BASIC. Daca nu ai matrici sau siruri de retinut in zona 'VARS' atunci poti folosi comanda 'LOAD "" ' compilata, in caz contrar vor apare erori si greseli stranii.

19. Viteza

Viteza este criteriul dupa care se orienteaza utilizatorii de compilatoare. Compilatorul COLT creaza un cod foarte rapid. Desi unele functii (DRAW, CIRCLE) cit si unele comenzi sint la fel de rapide compilate sau interpretate, celelalte (majoritare) functii si comenzi cistiga foarte mult in viteza sub compilatorul COLT. O bucla FOR .... NEXT aflata la capetii a 16K de program BASIC se executa de 800 ori (!) mai rapid sub COLT. In general este de asteptat o imbunatatire de 50 de ori a timpului de executie la programele uzuale.

Comenzile Executorului sint in mod considerabil mai rapide decit alte comenzi. O simpla comanda din Executor este mai rapida decit o comanda "sprite" tipica MOVE care compilata poate fi doar de 2-4 ori mai rapida. Comenzile Executorului sint intodeauna foarte eficiente.

In cadrul testelor din standardul 'KILOBAUD', avantajul COLT-ului fata de interpretorul Spectrum este de 75 de ori.

20. Erori mai ciudate

In majoritatea cazurilor cind COLT da rezultate stranii, problema poate fi cauzata de folosirea eronata a modului de lucru cu numere intregi. Aminteste-ti ca un calcul care ar trebui sa dea in mod normal un rezultat fractionar, va fi trecut de COLT intr-unul intreg si că domeniul maxim de numere cu care se poate opera este de la -32768 la 32767.

21. COLT in detaliu
- informatii tehnice

In acest capitol se dau detalii asupra modului de operare din COLT, a sarcinilor interne ale sistemului si căi de mărire a eficientei lor.

Compilatorul este chemat cu comanda :

```
RANDOMIZE USR 60000
```

Daca este gasită vreo instructie BASIC care nu poate fi compilata, linia pina la intructia respectiva este afisata urmata de cursorul '?' pilpiitor. Cursorul '>' va fi pus automat in dreptul liniei respective si astfel poate fi corectata cu ajutorul comenzii 'EDIT'.

Dupa primul pas de compilare, in timpul caruia se afişeaza orice eroare, se executa imediat si al doilea pas. In acest timp va fi afisat mesajul :

```
Compiling line xxxx
```

Dupa terminarea compilarii acest mesaj este urmat de un tabel care cuprinde informatii utile despre starea codului compilat in calculator :

```
Begin  End    Vars   Lines
xxxxx  xxxxx  xxxxx  xxxxx

Names  Nmtop  Ctop   Nvars
xxxxx  xxxxx  xxxxx  xxxx

To run type RANDOMIZE USR xxxxx
```

Pentru a explica aceste numere, se foloseste diagrama :

```
-------------------------------------------------------------------------
      I      I               I         I      I       I
BASIC I Spare I Compiled Code I Unused I Vars I Lines I Names
______I_______I_______________I________I______I_______I_________

   STKEND  RAMTOP            End      Vars   Lines   Names
           = Begin
__________________
     I  I
     I  I COLT
  ___I__I_________
     I  I
     I Ctop
     I
Nmtop
```

'STKEND' - Este luat din variabilele de sistem BASIC (adresa 23653 si 4) care marcheaza sfirsitul spatiului ocupat de BASIC.

'RAMTOP' - Este luat din variabilele de sistem BASIC (adresa 23730 si 1) care marcheaza ultima adresa ce poate fi ocupata de BASIC; este asezat prin 'CLAR n' si se afla in mod normal la adresa 40000.

'Begin ' - De obicei este egal cu 'RAMTOP', dar cind se compileaza 32 Ko, se schimba.

'End' - Macheaza sfirsitul codului compilat. De la 'End' pina la 'Vars' este loc liber pentru extinderea programului.

'Vars' - Aici sint stocate variabilele COLT. Variabilele folosite de COLT nu sint aceleasi cu cele din BASIC de acelasi nume. Spatiul necesar este de 2 octeti pentru o noua variabila dupa cea initiala. Toate variabilele au o intrare in acest tablou: 26 de variabile de o litera sint primele si contin spatiu pentru valori, bucle si pasi; urmatoarele 26 sint siruri si in acest caz tabloul contine indicatori pentru fiecare sir, lungimea sa si lungimea maxima absoluta; ultimile sint 26 de matrici cu indicatori similari.

'Lines' - Este un tablou de numere de linii si de adrese de salt construit in ordinea de compilare a 'GOTO' si 'GOSUB', etc. Pastreaza si estimeaza 'GOTO s'. Acest spatiu este de patru ori numarul liniilor.

'Names' - Poti avea pina la 255 de variabile in COLT. Primele doua sint variabile de sistem, urmatoarele 78 sint variabile de siruri, de matrici si variabile de bucla (de o singura litera). Acestea sint destinate sa faca codul cit se poate de rapid. Spatiul de la 'Names' la 'Ctop' este rezervat pentru numele variabilelor tale. In lipsa lor, sint alocate totusi citeva sute de octeti dar tu poti sa schimbi acest lucru (vezi mai departe).

'Nmtop' - Ultima locatie folosita intre 'Names' si 'Ctop' pentru numele variabilelor.

'Nvars' - Este numarul de nume de variabile pe care le folosesti. Nu va fi niciodata mai mic de 80.

'Ctop' - Ultima locatie care poate fi folosita in timpul compilării, Se află in mod normal inaintea compilatorului (sau a Executorului in caz că il folosesti). Acesta locatie poate fi schimbată (vazi mai departe). Inceputul compilatorului se afla la 59200 si al Executorului se afla la 52000.

22. Schimbari de spatii alocate - citeva adrese utile

Toate numerele sint stocate pe doi octeti in format normal Z80 (adica inversati). Adresele care prezinta interes sint:

59987 - Normal este zero, dar ea contine noul RAMTOP atunci cind se compileaza 32 Ko.

59991 - Conţine spatiul alocat numelor de variabile.

59993 - Contine adresa 'Ctop'. Poţi schimba această adresa dacă compilezi suroutine.

59997 - Foloseste 'RANDOMIZE USR 59997' pentru accesul COLT-ului la numerele aleatoare din BASIC.

23. Chemari intre BASIC si COLT

Dupa ce ai terminat programul si l-ai compilat, vei sterge tot BASIC-ul si vei folosi codul compilat ca o subrutina (o vei chema cu 'RANDOMIZE USR xxxxx'. Este o subrutina reala si variabilele folosite in COLT nu sint cele din BASIC cu acelasi nume. Cea mai rapida cale de a sterge BASIC-ul ete 'NEW'. Acesta sterge totul pina la RAMTOP. Codul compilat este deasupra RAMTOP-ului si e salvat de la stergere.

Acum tu poti chema codul compilat doar prin declaratia:

RANDOMIZE USR n

- unde 'n' este valoarea RAMTOP-ului aleasa de tine, de obicei 40000. Aceasta declaratie poate fi data imediat sau intr-o linie de program. In amindoua cazurile nici o alta declaratie ce urmeaza pe linie nu va fi executata.

Alta restrictie usoara este aceea ca in timp ce BASIC-ul poate chema oricit de multe sectii compilate (vezi 'Compilarea subrutinelor'), nu poti chema o sectie COLT de la alta, deoarece intrarea unei sectii compilate va sterge intodeauna sirurile si matricile COLT din spatiul 'Vars'. In plus, la sfirsitul sectiei compilate nu se afla o simpla reintoarcere (RET) in BASIC ci un salt la urmatoarea declaratie BASIC dupa cea care a chemat sectia compilata. Acest mod de lucru este necesar datorita pozitiei nesigure a stivei masinii dupa anumite rutine aflate in ROM. Vezi cap. '14. Revenirea in BASIC'. Din aceleasi motive nu poti chema codul compilat din propriul tau cod-masina.

24. Compilarea subrutinelor

Este posibil sa compilezi citeva sectii de BASIC pentru a crea subrutine total independente. Procedeul este urmatorul:

1) Compileaza prima sectie. Noteaza-ti valorile 'End' si 'Vars', care apar pe ecran dupa compilare. Scade valoarea cea mica din cea mare si modifica RAMTOP (prin 'CLEAR n') astfel ca sa crească cu rezultatul obtinut minus unu - adica Vars-End-1. De exemplu daca RAMTOP a fost 40000 (='Begin'), 'End'a fost 54000 si 'Vars' 57000, atunci 57000-54000-1=2999 si tu trebuie sa comanzi 'CLEAR 42999'.

2) Recompileaza cu noul RAMTOP si noteaza-ti noua valoare a lui 'Begin' care va apare pe ecran. Aceasta valoare va fi argumentul functiei USR atunci cind vei chema aceasta sectie compilata.

3) Modifica valorea 'Ctop', prin locatia 59993 si 4, astfel incit sa contină vechea valoare a lui RAMTOP (egala cu vechea valoare 'Begin') minus unu. Astfel se protejeaza sectia compilata generata anterior.

4) Incarca sau scrie urmatoarea sectie BASIC si introdu comanda 'CLEAR n' unde 'n' este locatia de la care vrei sa fie a doua sectie compilata.

5) Acum urmareste procedeul de la 1) la 4) de cite ori este nevoie.

6) Salveaza de la ultimul RAMTOP folosit pină la 65535 (aceasta va include tot UDGs). Este bine să faci acest pas dupa fiecare sectie compilata.

7) Desi pot fi variabile cu acelasi nume in sectii diferite, ele nu sint aceleasi variabile. Cind reintroduci o sectie de la inceput, toate variabilele sint zero.

25. Reintrarea codului compilat

Uneori este folositoare reintroducerea unei sectii compilate fara reinitializarea tuturor variabileleor, de ex. dupa o eroare sub Executor. Acest lucru este posibil daca reintroduci codul cu 19 locatii de la punctul de intrare. De exemplu, daca codul este la 40000, atunci

```
RANDOMIZE USR 40019
```

- nu va sterge matricile, sirurile sau variabilele. Pentru a fi totusi corect trebuie indeplinite citeva conditii:

1) Nu poti reintroduce codul - fara a sterge variabilele - daca ai folosit intre timp RUN sau CLEAR.

2) Trebuie sa intri in punctul normal al codului, inainte de a folosi alternativa "+19".

3) Nu trebuie sa-ti modifici BASIC-ul intre chemàri, prin adáugarea sau modificarea de linii sau prin deschiderea sau inchiderea de canale.

4) Poti reintroduce doar ultima sectie compilatǎ care a rulat.

Un exemplu:

```
 1 IF A=1 THEN GOTO 10
 2 DIM A(10)
 3 FOR K=1 TO 10 : LET A(K)=K
 4 NEXT K
10 LET A=1
```

Prima oara cind programul este rulat, liniile vor fi executate de la 1 la 10 (folosind 'RANDOMIZE USR n'). Daca programul este re-rulat cu 'RANDOMIZE USR (n+19)', atunci doar liniile 1 si 10 vor fi executate.

26. Executorul

Executorul face parte integral din programul COLT. El permite ca numeroase extra-comenzi sa fie folosite atit in BASIC Spectrum cit si in programele compilate de COLT. Executorul este incărcat si activat in acelasi timp cu compilatorul din COLT (el genereaza ceasul din coltul ecranului), dar poate fi oprit sau remutat daca spatiul lui este necesar pentru a fi folosit de compilator. Evident, dezactivarea sàu remutarea Executorului inseamnǎ cǎ nici o comanda de Executor nu trebuie sǎ aparǎ in programele BASIC interpretate sau compilate.

Posibilitǎtile Executorului sint:

* Grafice "sprite" pentru desene si obiecte sofisticate aflate in miscare.

* Functii de fereastrǎ, cu posibilitati de definire a mai multor ferestre, inclusiv "scroll" individual de fereastra si atribute de cadru.

* Generare pe ecran de "user-defined graphics".

* Programele in cod-masina si subrutinele pot fi chemate cu parametrii.

* Declaratia ON ERROR GOTO s .

* PEEK si POKE pe doi octeti.

* Transformǎri din zecimal in binar sau hexazecimal si din hexazecimal in zecimal.

* Evalueazǎ memoria rǎmasǎ liberǎ.

* Chemarile nu mai trebuie facute cu 'USR' odata ce Executorul a pornit. Fiecare comanda de Executor seamana cu BASIC-ul si controlul de sintaxa functioneaza similar.

* Toate comenzile Executorului pot fi date imediat sau amestecate printre declaratile BASIC dintr-un program.

* Toate comenzile Executorului pot fi compilate de COLT.

* Redefinirea comenzilor BASIC

* Sterge linii de program in bloc

* Compatibilitate totala cu microdrive

* Programare de taste - ideal pentru comenzi microdrive

* Ceasul de pe ecran cu facilitati de oprire si pornire

* Se afiseaza numarul liniei ce este executata

* Taste cu functie de comanda pentru pornirea compilatorului

* Taste cu functie de comanda pentru rularea programului de la RAMTOP (sub COLT)

* Tasta pentru repetarea ultimei comenzi

27. Utilizarea Executorului

Codul Executorului incepe de la adresa 52470 si se sfirseste la 59200. Daca vrei sa salvezi codul care include si Executorul atunci el trebuie oprit intii, altfel nu-l vei putea verifica.

Daca nu doresti sa utilizezi Executorul si ai nevoie de spatiul pe care il ocupa, atunci urmatoarea comanda

RANDOMIZE USR 59190

- il va sterge permitind compilarea urmatorilor 7K din BASIC. In cazul in care ai facut acest lucru, codul nu trebuie să cheme nici o rutină a Executorului.

Executorul utilizeaza modul 2 de intreruperi, asa ca nu poate fi folosit cu alte programe care folosesc intreruperi. El este inactivat (dar nu sters) prin 'NEW', dar daca trebuie sa faci 'NEW' fara a opri Executorul atunci exisată comanda:

* fx d 1,9999: CLEAR

- care are acelasi efect ca si 'NEW', dar fara a se atinge de Executor.

Executorul va fi pornit cu:

RANDOMIZE USR 55020

si va fi oprit cu:

RANDOMIZE USR 55010

Programele care folosesc Executorul vor rula cu 10% mai lent decit cele care nu-l utilizeaza, datorita intreruperilor.

28. Comenzi cu acces rapid

Comenzile cu acces rapid permit folosirea anumitor comenzi ale Executorului cu un numar foarte mic de apasari pe taste. In tabelul care urmeaza, simbolurile 'E!' inseamna 'apasa SPACE si apoi litera, elibereaza SPACE si apoi litera'. Nu este nevoie sa apesi ENTER.

E!t	- afiseaza ceasul
E!l	- afiseaza numarul liniei executate ("trace on")
E!o	- stege ceasul sau afisarea liniei executate ("trace off")
E!s	- opreste ceasul (care ramine afisat) - ceasul merge in continuare in interiorul computerului
E!z	- ceasul este pornit de la zero
E!i	- initializeaza ceasul astfel: se apasa numarul dorit pentru primul digit al ceasului, apoi ENTER pentru a trece la urmatorul digit
E!r	- se readuce ultima comanda pentru a fi modificata si relansata
E!c	- cheama compilatorul COLT (inlocuieste comanda 'RANDOMIZE USR 60000')
E!p	- executa programul de la RAMTOP+1
E!k	- salveaza codul; astfel se inregistreaza pe banda codul compilat, compilatorul si UDGs; comanda poate salva si pe microdrive; copia salvata pe banda poate fi reincarcata cu 'CLEAR n:LOAD "name" CODE' unde 'n' este valoarea din 'RAMTOP' de la care codul a fost compilat; aceasta va fi 40000 daca nu ai schimbat-o cumva; retine ca atunci cind codul este reincarcat, compilatorul este dezactivat, iar 'E!c' nu trebuie folosit; nu intrerupe salvarea cu BREAK altfel compilatorul va fi stricat
E!x	- intrerupe si scoate afara aproape orice program; deasemenea readuce ecranul la hirtie alba si cerneala neagra

29. Utilizare - definire 'soft' (functii) de taste

Comenzile pentru microdrive sint in mod normal greoaie. Tastele programabile -'soft'- permise de Executor, iti ofera posibilitatea alegerii expresiilor proprii pentru comenzi din doua apasari de taste.

Sint noua taste de utilizare - definire ('1' la '9'). Executia comenzilor este simpla, prin apasarea pe 'E! numar'.

Iata cum se pot defini tastele: trebuie sa ai o linie

1 REM

- in care sa tiparesti comanda sau caracterele pe care vrei sa le introduci prin aceasta facilitate; poate include comenzi ca THEN,IF,PRINT, etc. (ai nevoie de ':'naintea comenzii); fiecare definire se separa cu semnul '^' sau '!'; daca folosesti '^'atunci 'ENTER' se produce automat; daca folosesti '!' atunci comanda respective se comporta ca un fel de input, permitindu-ti sa faci modificari inainte de a apasa 'ENTER'.

Exemplu:

```
1 REM : PRINT "Mary has a little lamb" ! : PRINT "its
fleece was white as snow" ^
```

- apasind 'E!1' va apare

PRINT "Mary has a little lamb"

- ti se permite sa faci modificari, iar apoi apasind 'ENTER' comanda se executa. Apasind 'E!2', se va tipari imediat:

its fleece was white as snow

30. Extinderea comenzilor BASIC

Comenzile Executorului se introduc numai daca Executorul este pornit si sint asemanatoare cu cele din BASIC. Ele se pot amesteca in program cu celelalte comenzi BASIC.

Forma generala este urmatoarea:

* fx token / letter argument 1, argument 2,....

Fiecare argument este o expresie validata si pot fi maximum 16 argumente. Excesul de argumente va fi ignorat si nu constituie o eroare. Argumentele sint tratate ca numere intregi.

'token / letter' pot fi declaratii BASIC 'FN','FOR','ERASE' sau litere mari sau mici (literele din paranteze se vor omite, ele avind doar un rol explicativ).

Rezultatul unei comenzi de Executor este in unele cazuri un numar intreg pe 8 biti (pina la valoarea 255), iar in alte cazuri un numar intreg pe 16 biti.

31. Comenzi de fereastra

Executorul iti permite se definesti mai multe ferestre pe ecran, care pot fi mai mici decit 32 de coloane pe 24 de linii. Pentru interiorul acestor ferestre exista niste functii speciale

* fx FN a,b,c,d
  - defineste o fereastra ('FN' vine de la FeNetre !) care are punctul de referinta in coltul sting la linia 'a' si coloana 'b', este adinca de 'c' linii si lunga de 'd' coloane, iar fereastra minima este de 2x2

* fx PAPER n
  - schimba culoarea hirtiei din fereastra definita mai inainte, 'n' luind valori de la 0 la 7; nu este o eroare daca 'n' se afla in afara acestui domeniu findca 8 va deveni 0, 9 va fi 1, 10 va fi 2, etc

* fx PAPER n,1
  - schimba culoarea hirtiei doar pentru acele locatii de caractere care au atributele curente de hirtie

* fx INK n
  - idem 'PAPER n'

* fx INK n,1
  - idem 'PAPER n,1' dar cu atribute 'INK'

* fx FLASH 0
  - inceteaza flash-ul pe toată fereastra

* fx FLASH n
  - flash pe toata fereastra (pentru n>0)

* fx FLASH n,1
  - idem 'INK n,1' dar cu atribute 'FLASH'

* fx BRIGHT 0
  - idem 'FLASH 0'

* fx BRIGHT n
  - idem 'FLASH n'

* fx BRIGHT n,1
  - idem 'FLASH n,1' dar cu atribute 'BRIGHT'

* fx INVERSE 0
  - inverseaza atributele ferestrei (se inverseaza culoarea hirtiei cu cea a cernelii)

* fx INVERSE 0,1
  - idem 'FLASH n,1'

* fx SCREEN$ n,p,q
  - muta cerneala de pe ferastra definita anterior, in directia 'n' (5,6,7,8), cu un pixel de 'p' ori, iar 'q' defineste felul deplasarii:
    - q=2 rásucirea ferestrei dintr-o parte in alta
    - q=1 umple partea deplasata in culoarea cernelii
    - q=0 umple partea deplasata in culoarea hirtiei
    - q=-1 rasucire dintr-o parte in alta cu inversare

* fx ATTR n
  - muta atributele de fereastra in directia 'n'(5,6,7,8) iar in zona deplasata apar atributele curente

* fx LINE n,p,r,q
  - muta in directia 'n'(numai 5 si 8), cu un pixel de 'p' ori, caracterele de pe linia 'r'; rotirea este definita de 'q' la fel ca la 'SREEN$'; linia 'r' trebuie se fie in fereastra definita

32. Comenzi "sprite"

Comenzile "sprite" dau o noua dimensiune posibilitătilor grafice ale Spectrum-ului. Cu ajutorul lor se pot defini desene de pina la 32x24 de pixeli care pot fi pozitionate oriunde pe ecran. Cind unul dintre acestea este mutat, se reinstaleaza fondul care era inainte de definirea desenului. Pot exista pina la 16 de astfel de desene, ele fiind vizibile sau nu, dupa voie.

Exista anumite restrictii. Nu trebuie efectuat "scroll" asupra zonei de ecran unde se gaseste un desen daca doresti sa-l muţi ulterior.

* fx FORMAT n,a,b,c,d,m,e
  - 'n' este numarul desenului (de la 1 la 16), care este definit la pozitia (a,b) cu dimensiunile (c,d), parametrii a,b,c,d fiind pixeli; definirea desenului

incepe de la adresa 'm' de la stinga la dreapta si de sus in jos; un astfel de desen este intodeauna vizibil dupa ce a fost definit, culoarea cernelii fiind cea din zona in care a fost definit desenul, iar culoarea hirtiei este '9'(contrast); pentru a ne ajuta in generarea desenelor si a UDGs in general, este disponibil un sistem de definire daca parametrul 'e=1', atunci urmatoarele taste au anumite roluri:

(0) sterge pixelul curent
(1) marcheaza pixelul curent
(3) sterge tot desenul
(4) marcheaza tot desenul
(5) stinga
(6) jos
(7) sus
(8) dreapta
(9) readuce desenul la forma initiala
(ENTER) se intoarce la urmatoarea comanda
(CAPS-SHIFT) arata desenul la pozitia (a,b) si la dimensiunile definite

De exemplu, definim un UDG simplu (litera 'A') folosind o comanda asemanatoare cu aceasta:

* fx FORMAT 1,100,100,8,8,65368,USR "A"

Retine ca definirea unor astfel de desene va opri ceasul, respectiv "line trace".

* fx MOVE n,a,b,e
- muta desenul 'n' la pozitia (a,b); daca este prezent parametrul 'e' atunci adresa desenului este mutata la 'e'; aceasta iti permite sa alternezi UDGs-urile pentru a anima desenul; cind un desen este mutat, el devine intotdeauna vizibil daca ramine pe ecran

* fx ERASE n
- sterge temporar desenul 'n', dar il lasa definit

* fx ERASE
- sterge temporar toate desenele desi ele ramin definite pe ecran; aceasta operatie trebuie sa fie precedata de 'CLS' deoarece urmatorul 'MOVE' va redesena pe ecran imaginea care tocmai a fost stearsa

* fx VERIFY n
- aceasta comanda evita suprapunerile; ea verifica daca desenul 'n' se suprapune peste alt desen si face modificari conform locatiei 54983; daca locatia este 0 dupa un 'VERIFY' inseamna ca desenul 'n' nu se suprapune peste nici un alt desen; altfel locatia contine numarul desenului care este atins.

* fx RESTORE n
- readuce desenul 'n' care a fost sters

Pentru a folosi "sprite" (desenele) se da prima oara un FORMAT pentru fiecare, folosind modul de definire daca este necesar. Astfel se face o identificare a fiecarui desen si da o pozitie initiala. Dupa FORMAT-uri se pot da MOVE, ERASE si RESTORE dupa preferinta.

33. Citirea tastaturii

'INKEY$' citeste corect tastatura dar numai pentru o tasta apasata, acest lucru nefiind suficient pentru jocuri. Este mai utila folosirea comenzii 'IN'. De exemplu daca 'IN 254' nu este egal cu 255, atunci inseamna ca una sau mai multe taste sint apasate. Daca 'IN 65278' nu este egal cu 255 atunci inseamna ca una sau mai multe taste dintre (CAPS-SHIFT),Z,X,C,V sint apăsate (vezi manualul de BASIC Spectrum cap. 23)

34. Alte comenzi

Literele din paranteze nu fac parte din sintaxa comenzii, ele au doar un rol explicativ:

* fx T(ime)
- porneste afisarea ceasului; acesta se opreste in timpul 'BEEP' si in timpul operatiilor cu casetofonul; valoarea ceasului poate fi culeasa cu 'PEEK' din BASIC de la locatiile 54986, 7, 8 si 9

* fx L(ine)
- activeaza "line trace"; numarul liniei in curs de executie este afisat in partea dreapta a ecranului; daca se utilizeaza aceasta comanda impreuna cu REM #2 acest numar va fi vizibil; el va fi actualizat la 1/50 secunde

* fx O(ff)
- dezactiveaza afisarea ceasului sau "line trace"; ceasul merge incontinuare in Executor

* fx Z(ero)
- porneste ceasul de la zero

* fx I(nitalize)
- initializeaza ceasul; comanda are sintaxa:
* fx I h,m,s

* fx S(top)
- opreste ceasul sau linia, afisind valoarea la care s-a oprit; (ceasul continua sa mearga in Executor)

* fx F(ree)
- tipareste spatiul disponibil intre 'STKEND' si 'RAMTOP'

* fx P(rog)
- lanseaza programul de la RAMTOP+1

* fx LEN
  - tipareste ultima adresa a programului BASIC (este valoarea 'STKEND')

* fx USR m,a,b,c ...
  - se lanseaza rutina de la adresa 'm'; regisrul HL este pozitionat pe adresa 'a' (2 octeti); aceasta inseamna ca putem trimite parametrii la o subrutina in cod masina

* fx BIN n
  - tipareste 'n' (numar zecimal) in binar pe 16 biti

* fx CODE n
  - tipareste valoarea hexa a numarului zecimal 'n'

* fx H(ex)
  - intra in modul 'INPUT' si primeste un numar hexa; dupa ce se tasteaza <ENTER> se tipareste echivalentul zecimal; se ignora caracterela ilegale iar strgerea nu este permisa

* fx D(elete) m,n
  - sterge liniile de program intre 'm' si 'n'

* fx W(here) n
  - tipareste locatia de memorie corespunzatoare liniei de program 'n'; se da adresa octetului cel mai semnificativ al numarului de linie

* fx G(o to) n,m
  - daca apare o eroare 'm' executia continua de la linia n; daca 'm'=0 sau nu apare atunci 'BREAK' (eroarea 'D' sau 'L') va da totusi un raport de eroare; daca 'm'=1 atunci se detecteaza 'L' iar daca 'm'=2 atunci se detecteaza ambele (periculos pentru ca include E!E!); daca aceasta comanda se afla in codul compilat ea transfera comanda unei linii 'n' din programul BASIC si nu unei linii din programul compilat; daca 'n'=0 detectarea erorilor este oprita; numarul erorii este stocat la adresa 54982 inainte de a fi controlata si de saltul la linia 'n'; aceasta comanda este compatibila cu erorile de la interfata 1

* fx STOP n
  - stop cu raport 'n'; aceasta va fi detectata de '* fx GO TO' de deasupra numai daca 'n' nu este 0

* fx DATA n,m
  - este un POKE pe doi octeti; 'n' este un numar intreg pe 16 biti plasat in locatiile m, m+1

* fx PEEK m
  - este un PEEK pe doi octeti; se returneaza un intreg pe 16 biti fara semn

* fx LN n,m,p
  - furnizeaza o functie "loggins" care te ajuta sa vezi care parte a programului (fie BASIC fie cod masina) consuma cel mai mult timp, astfel ca tu poti sa-ti concentrezi efortul asupra celui mai mic procentaj de cod care consuma cel mai mult procentaj de timp; in general maximum este 10% de cod consuma 90% din timp;

  - pentru BASIC parametrul 'm' este linia de start, 'n' este incrementul lui 'm' si 'p'=2; se returneaza 10 numere: primul contine timpul consumat pina la linia 'm' (50 de unitati reprezinta o secunda), al doilea este timpul consumat de la 'm+1' la 'm+n', al treilea este timpul de la 'm+n+1' la 'm+2n, al patrulea este timpul de la 'm+2n+1' la 'm+3n', ..., iar ultimul numar reprezinta timpul consumat de la 'm+8n+1' pina la sfirsitul programului;

  Ex pt BASIC:
  *fx LN 100,10,2

  - pentru cod masina parametrul 'm' reprezinta adresa de inceput, 'n' pasul si 'p'=1;

  - daca parametrul 'p'=0, facilitatea este dezactivata; o noua instructiune '* fx LN' sterge datele unei '* fx LN' anterioare;

* fx LLIST
  - face ca datele furnizate de functia de mai sus sa fie tiparite pe ecran ca o tabela; aceasta comanda nu opreste sau reseteaza procesul "logging"

35. Redefinirea functiilor BASIC

Putem defini propriile noastre instructiuni sau putem redefini functiile Executorului. Adresa codului masina al rutinei noastre (in forma de 'JP adresa') trebuie plasata in locatiile 55001,2 si 3. 'JP' este 'C3' in hexazecimal. Rutina ta va plasa cuvintul cheie sau prima litera de dupa '*fx' in acumulator. Registrele pare HL pastreaza adresa primului argument (pe doi octeti). Daca bitul 7 al variabilei de sistem 'FLAGS' este zero atunci sintem in decursul verificarii sintactice si trebuie returnat zero in acumulator (fanionul Z va fi si el pus pe zero) daca este vorba de caracterul sau cuvintul utilizatorului. In caz contrar se face revenirea fara sa se intimple nimic. Daca bitul 7 a lui 'FLAGS' este 1, se executa un program BASIC si trebuie lansat codul nostru. La terminarea acestuia trebuie pus acumulatorul pe zero.

De exemplu, pentru a definii: '* fx SQR' - o functie care sa tipareasca un caracter, trebuie folosit urmatorul cod:

```
CP 187              ; 187 este codul cuvintului cheie 'SQR'
RET NZ              ; daca nu este SQR se revine
XOR A               ; pune pe zero registrul A si fanionul Z
BIT 7,(IY+1)        ; testeaza bitul 7 de la 'FLAGS'
RET Z               ; revine daca este zero
LD A,(HL)           ; plaseaza primul argument in A
RST 16              ; tipareste caracterul curent via ROM
XOR A               ; pune pe zero acumulatorul
RET                 ; revine
```

Pentru a tiparii litera 'A' (care are valoarea 65 in ASCII) se da comanda '* fx SQR 65'.

36. Mesajele de eroare ale Executorului

Instructiile Executorului se verifica sintactic in mod obisnuit (cu un cursor pilpiitor '?' in caz de eroare). In cazul erorilor de program mesajele sint urmatoarele:

Q Parameter error
- au fost folositi mai mult de 16 parametrii

P FN without DF
- o comanda '*fx' se executa din codul compilat; aceasta eroare nu se poate intimpla decit in cazul in care versiunile COLT si Executor nu sint compatibile

A Invalid argument
- numerele de linii utilizate in comanda 'DELETE' sint incorecte

B Integer out of range
- toate argumentele trebuie sa fie intregi si cuprinse intre -32768 si 32767; acesta eroare mai apare si atunci cind se defineste o fereastra prea mare sau se manipuleaza cu linii din afara ecranului

5 Out of screen
- ecranul nu este suficient de mare pentru comanda 'FORMAT'

Executorul foloseste modul doi de intreruperi pentru a lucra cu ceasul, "line trace", etc., si sint actualizate de 50 de ori pe secunda. Variabila de sistem 'FRAMES' (23672, 3 si 4) este modificata de acest proces. Programele care folosesc aceste facilitati vor rula cu 10% pina la 15% mai incet. Ceasul se opreste in timpul BEEP-ului, in timpul comenzilor de incarcare si salvare de pe caseta, microdrive si alte comenzi care blocheaza sistemul de intreruperi.

Traducere dupa:

COLT ZX BASIC Compiler Manual

Published by Hisoft
180 High Street North
Dunstable LU6 1AT

First Edition August 1985

BETA BASIC VI 8

1. INTRODUCERE

BETA BASIC (BB) adaugă 30 de noi comeñi şi mai mult de 20 npi funcţii acelora existente în SPECTRUM BASIC (SB); în plus, unele comenzi au fost considerabil extinse şi o serie de facilităţi au fost adăugate, cum ar fi: un cursor clipitor (flash) al liniei curente, un BREAK mai puternic şi posibilitatea de a muta cursorul de editare în toate direcţiile.

Noile comenzi sînt obţinute prin trecerea în modul "GRAPHICS" şi apăsarea unei chei, eventual cu SHIFT; noile funcţii se obţin introducînd cuvîntul cheie FN urmat de o literă şi "(" sau "$".

BB constă într-un program în cod maşină de 1,3 kiliocteţi (RAMTOP este coborît la 55800 pentru a proteja codul BB), precum şi din liniile 0, 1, 2, în BASIC, liniile 1 şi 2 se şterg după încărcarea codûlui maşină; linia 1 poate fi utilizată pentru salvarea BB (CLEAR ramtop ; LOAD " " CODE) ; linia 0 conţine definiţiile pentru noile funcţii ale BB şi ea rămîne prezentă tot timpul, deşi nu e listată. Din acest motiv, nu trebuie încărcate cu LOAD programe care nu au fost scrise cu BB (a se utiliza MERGE care nu disting linia 0).

Utilizarea BB va conduce la o viteză mult mai mare de execuţie a programelor, în special a celor lungi, care utilizează numeroase instrucţii GOTO şi GOSUB.

NEW e mai puţin radical ca de obicei, şterge orice program cu excepţia liniei 0 şi execută un CLEAR.

Cu excepţia uneia (EDIT), celelalte comenzi se obţin prin trecerea în modul grafic.

2. DESCRIEREA COMENZILOR BETA BASIC

2.1. ALTER

- sintaxa; ALTER (descriere de atribut) TO descriere de atribut.
- tasta; A

ALTER permite manipularea fişierului care conţine atributele ecranului (informaţiile INK, PAPER, FLASH şi BRIGHT, pentru fiecare caracter.

În cea mai simplă formă a sa, ALTER poate modifica culoarea INK sau PAPER a întregului ecran, fără a-l şterge:

```
100 PRINT AT 10 ; "TEST":PAUSE 50 ; ALTER TO PAPER 1
```

sau

```
..... ALTER TO PAPER 2,INK 7, FLASH 1
```

De asemenea, putem fi selectivi în ceea ce priveşte alegerea poziţiilor de caractere care sînt efectuate de comandă, incluzînd o descriere a atributelor acestora înaintea lui TO, astfel:

```
ALTER INK 7 TO INK 0
```

va schimba tot ceea ce e scris cu cerneală albă în cerneală neagră, sau

```
ALTER INK 3, BRIGHT 1, PAPER 7 TO INK 5, FLASH 1
```

2.2. AUTO

- sintaxa : AUTO (număr linie)(,pas)
- tasta : 6

AUTO reprezintă o facilitate de numerotare automată a liniilor la introducerea programelor; dacă nu se specifică pasul, se va folosi pasul 10; dacă se tastează doar AUTO, atunci pe linia de editare va apare numărul de linie corespunzător poziţiei curente a cursorului + 10.

AUTO devine neoperaţional cînd numărul liniei este mai mic decît 10 sau mai mare decît 9983, sau la tipărirea unui mesaj de eroare; o modalitate convenabilă de a ieşi din AUTO este să se apese BREAK mai mult de o secundă; dacă dorim să sărim un bloc de linii în timp ce utilizăm AUTO, trebuie şters numărul de linie ce apare automat şi tastat numărul de linie la care dorim să sărim.

2.3. BREAK

- taste: (shift)+(space)

Este mai puternic decît în cazul SB, deoarece realizează oprirea programului şi în cazul cînd se execută o rutină în cod maşină, nu numai un program BASIC.

2.4. CLOCK

- sintaxa : CLOCK număr sau string
- tasta : (C)

CLOCK furnizează în colţul din dreapta sus al ecranului timpul în ore, minute şi secunde; în funcţie de modul de lucru ales, această comandă oferă următoarele facilităţi :

MODE	ALLARM GOSUB	AUDIBILE ALARM	DISPLAY
0	NO	OFF	OFF
1	NO	OFF	ON
2	NO	SET	OFF
3	NO	SET	ON
4	YES	OFF	OFF
5	YES	OFF	ON
6	YES	SET	OFF
7	YES	SET	ON

Ceasul va porni în momentul încărcării BB de la valoarea " 00 : 00 : 00" ; pentru a-l vizualiza se va da comanda CLOCK 1 ; pentru a-l fixa pe o valoare convenabilă vom tasta

```
CLOCK " 23 : 23 : 23 " sau CLOCK " 23 / 23 / 23 "
(Clock "          10 " = CLOCK " 10 : 10 : 10 " )
```

De asemenea, se poate obţine un semnal sonor sau lansa în execuţie o subrutină la atingerea momentului de timp specificat.

2.5. CODURILE PENTRU DEPLASAREA CURSORULUI

CHR$ 8 - stînga
CHR$ 9 - dreapta
CHR$ 10 - jos
CHR$ 11 - sus

BB permite ca aceste caractere să fie tipăribile; instrucţia

```
PRINT CUR$ 10
```

va deplasa cursorul de tipărire cu o poziţie în jos.

2.6. DEF KEY

- sintaxa : DEF KEY one letter string; string
- taste : (1) + (shift)

DEF KEY redefineşte tasta specificată de "one letter string" astfel :

```
DEF KEY " 1 " ; " Hello "
```

Dacă se apasă (simbol shift)+(space), cursorul se va schimba într-un asterisc clipitor; dacă acum apăsăm (1), va apare scris " Hello ! " în partea de jos a ecranului.

Toate definiţiile se salvează împreună cu BB (sînt plasate deasupra RAMTOP) ; RAMTOP va fi coborît automat pentru a face loc definiţiilor.

2.7. DEF PROC

- sintaxa : DEF PROC nume de procedură
- tasta : (1) (la fel ca şi DEF FN)

DEF PROC marchează începutul codului unei proceduri cu numele "nume procedură"; O procedură poate avea acelaşi nume cu o variabilă, fără ca aceasta să creeze confuzii.

2.8. DELETE

- sintaxa : DELETE (număr linie) TO (număr linie)
- taste : (7)(aceeaşi ca şi ERASE)

DELETE şterge toate liniile din blocul specificat; dacă se omite primul număr de linie, se începe cu prima linie după linia 0, iar dacă se omite al doilea, se şterge pînă la sfîrşitul programului; DELETE TO diferă de NEW prin aceea că nu execută CLEAR asupra variabilelor.

2.9. DO

- sintaxa : DO WHILE condiţie
  DO UNTIL condiţie
  DO
- tasta : (D)

DO marchează începutul unui ciclu al cărui sfîrşit este marcat de LOOP ; este permisă suprapunerea ciclurilor DO - LOOP ; ieşirea din mijlocul unui ciclu DO - LOOP se poate face doar utilizînd EXIT IF sau POP ; în caz contrar vor apare erori în stivă.

2.10. EDIT

- sintaxa : EDIT număr linie
- tasta : (0)(fără modul grafic)

EDIT poziţionează cursorul pe linia specificată; aceasta va fi coborîtă în partea de jos a ecranului, pentru a fi editată.

2.11. DROKE

- sintaxa : DROKE adresa, număr
- tasta : (P)

DROKE înseamnă "double POKE"; echivalentul SB este :

```
POKE adresa, număr - INT(număr/256)-256
POKE adresa +1, INT(număr/256)
```

2.12. ELSE

- sintaxa : ELSE (instrucţie)
- tasta : (E)

ELSE este o parte a instrucţiunii IF-THEN-ELSE ; întotdeauna ELSE se referă la IF-THEN imediat precedent.

2.13. END PROC

- tasta : (3)

END PROC marchează sfîrşitul codului unei proceduri; la apelarea procedurii respective se va executa secvenţa dintre DEF PROC şi END PROC ; la întîlnirea END PROC se va face salt la instrucţia următoare apelului de procedură.

2.14. EXIT IF

- sintaxa : EXIT IF condiţie
- tasta : (1)

Această instrucţiune poate apare în interiorul unui ciclu DO - LOOP ; dacă condiţia respectivă este adevărată, se părăseşte ciclul, efectuîndu-se un salt la instrucţia care urmează după LOOP.

2.15. FILL

- sintaxa : FILL x,y
  FILL (INK colour); x,y
  FILL (PAPER colour) x,y
- tasta : (F)

FILL umple o zonă de PAPER cu INK dacă se utilizează FILL sau FILL (INK), sau umple o zonă de INK cu PAPER dacă se utilizează FILL (PAPER), începînd cu punctele de coordonate x,y; dacă punctul de coordonate x,y şi zonele înconjurătoare sînt INK şi noi utilizăm FILL (INK), nu se va întîmpla nimic; este corectă forma :

```
FILL PAPER ; x,y
```

Exemplu de utilizare :

```
10 FOR N = 1 TO 6
20 CLS
30 CIRCLE INK N ; 128,88,N * 10
40 FILL INK N ; 128,88
50 NEXT N
```

Este posibilă şi utilizarea unei forme complexe a lui FILL :

```
FILL INK 2 ; PAPER 1 ; FLASH 1 ; x,y
```

În acest caz, primul cuvînt de după FILL determină utilizarea culorii INK sau PAPER, iar următoarele modifică caracteristicile zonelor umplute.

Numărul de pixeli umpluţi în cadrul ultimei utilizări a lui FILL se poate obţine utilizînd funcţia FILL().

2.16. GET

- sintaxa : GET valabilă numerică sau valorică string
- tasta : (G)

La fel ca şi INKEY$, GET reprezintă un mod de a citi tastura ; diferenţa constă în aceea că GET aşteaptă apăsarea unei chei înainte de a continua; dacă se utilizează o variabilă string

tastăm (1) =) " 1 "
tastăm (A) =) " A "

iar dacă se utilizează o variabilă numerică

tastăm (1) =) 1
tastăm (9) =) 9
tastăm (A) =) 10
tastăm (B) =) 11
....

2.17. JOIN

- sintaxa : JOIN (număr linie)
- tasta : (shift)+(6)

JOIN are drept efect trecerea pe aceeaşi linie a liniei specificate (linia curentă indică nu se specifică nimic) şi a lniei următoare (dacă există).

Linia următoare îşi va pierde numărul şi va fi separată de linia precedentă prin " : " ; JOIN poate fi utilizat după SPLIT dacă numai o parte a unei linii şi să o adăugăm la alta.

2.18. KEY IN

- sintaxa : KEY IN string
- tasta : (shift)+(4)

KEY IN introduce "string" în program ca şi cum ar fi introduş de la tastatură; aceasta permite programelor să se poată automodifica; de exemplu, se pot genera automat instrucţii DATA :

```
10 LET A$ = 100 DATA " ; REM use DATA keyword
20 FOR N=0 TO 9
30 LET A$ = STR$(PEEK N)+","
40 NEXT N
50 LET A$ = A$(I TO LET A$ - 1);REM shap off last comma
60 KEY IN A$
```

KEY IN se poate folosi, de exemplu, pentru a scrie un editor în mod ecran în BASIC, se poate utiliza doar din program, tastat ca şi comandă va genera eroare.

2.19. KEYWORDS

- sintaxa : KEYWORDS 1 sau
  KEYWORDS 0

- tasta : (8)
Comanda KEYWORDS 0 dezactivează, iar KEYWORDS 1 activează comenzile BB (se dezactivează pentru a putea lucra cu caracterele grafice).

2.20. LIST

- sintaxa : LIST număr linie TO număr linie ; această sintaxă nu este permisă în SB ; la fel şi în cazul LIST.

2.21. LOOP

- sintaxa : LOOP WHILE condiţie
  LOOP UNTIL condiţie
- tasta : (L)

LOOP este o componentă a structurii DO - LOOP ; LOOP simplu cauzează saltul la instrucţia DO corespunzătoare ; LOOP WHILE şi LOOP UNTIL realizează condiţionarea saltului.

2.22. ON

- sintaxa : ON sau GOSUB ON număr; număr linie 1; număr linie 2; .....
- tasta : (O)

ON permite saltul la o anumită linie din lista de linii care urmează, în funcţie de valoarea expresiei numerice de după ON; se poate utiliza pentru implementarea elegantă a programelor interactive de tip meniu.

2.23. ERROR

- sintaxa : ON ERROR număr linie
- tasta : (N)

După execuţia comenzii ON ERROR se va face un GOSUB la numărul de linie specificat la apariţia oricărei erori; în timpul execuţiei acestei rutine, ON ERROR este dezactivat şi se reactivează la execuţia RETURN; dezactivarea permanentă se face cu ON ERROR 0. În cadrul rutinei de trasare a erorilor se pot utiliza variabilele sistem LINE, STAT şi ERROR.

2.24. PLOT

- sintaxa : PLOT coordonata X, coordonata Y (: string)

BB permite utilizarea PLOT nu numai la nivel de pixeli ci şi la nivel de caractere; în şirul de caractere pot fi incluse şi caracterele de control ale cursorului CHR$ 8 - 11 ; un astfel de plot se poate utiliza pentru obţinerea "word-wrapping-ului" la un editor în mod ecran.

2.25. POKE

- sintaxa : POKE adresa, string

BB permite realizarea POKE şi cu string-uri, nu numai cu numere; această comandă, împreună cu funcţia MEMORY$, permite manipularea rapidă a unor zone mari de memorie.

```
10 POKE adresa, MEMORY$()(n1 TO n2)
```

2.26. POP

- sintaxa : POP (variabila numerică)
- tasta : (Q)

POP extrage o adresă din stiva GOSUB/DO-LOOP/PROC; numărul de linie extras din stivă va fi atribuit variabilei care apare în comandă, dacă aceasta există ; în acest mod o procedură poate afla de unde a fost apelată; pentru revenirea dintr-o procedură unde a fost făcut un POP se foloseşte în loc de RETURN :

GOTO variabila +1

2.27. PROC

- sintaxa : PROC nume procedură
- tasta : (2) (aceeaşi ca şi la FN)

PROC realizează apelul procedurii definite cu numele respectiv; diferenţa între PROC şi GOSUB este aceea că apelul se face după un identificator şi nu după un număr de linie; la proceduri nu există parametri formali, nici variabile locale.

2.28 RENUM

- sintaxa : RENUM (start TO finish)(LINE new start)(STEP step)
- tasta : (4)

Această comandă renumerotează liniile începînd cu linia cu numărul "start" pînă la linia cu numărul "finish" ; optimal, prima linie poate primi valoarea "new start", iar pasul de numerotare poate fi fixat la "step".

Dacă trebuie numerotate linii de forma

GOTO EXPRESIE

acest salt poate avea loc la o linie care poate fi renumerotată ea însăşi; în acest caz, se abandonează renumerotarea cu raportul :

Y " Too hard "

2.29. ROLL

- sintaxa : ROLL direction code (,pixels)(;x,y;with,length)
- tasta : (R)

ROLL realizează mutarea întregului ecran sau a unui ecran nedefinit (fereastră) în sus, la stînga sau la dreapta; el nu distruge nimic din imaginea ce se găseşte pe ecran, doar o rearanjează.

Sintaxa ROLL este relativ complicată, dar aceasta este necesară doar în scopul de a defini fereastra care va fi deplasată; deplasarea întregului ecran cu un pixel se face cu

ROLL direction code

În funcţie de codul de direcţie se pot muta şi/sau atributele de culoare.

direction code	direction	applies to
1	LEFT	ATTRIBUTES
2	DOWN	-"-
3	UP	-"-
4	RIGHT	-"-
5	LEFT	PATERN DATA
6	DOWN	-"-
7	UP	-"-
8	RIGHT	-"-
9	LEFT	BOTH
10	DOWN	-"-
11	UP	-"-
12	RIGHT	-"-

Pentru a deplasa doar o fereastră în ecran, codul direcţie trebuie să fie de 4 parametri : coordonatele X şi Y ale colţului din dreapta sus al ferestrei, lărgimea ferestrei (în caractere, nu în pixeli) şi lungimea ferestrei, în pixeli ; această facilitate poate fi foarte utilă la jocuri.

Exemple ;

```
100 LIST : LIST : LIST
110 LET pixels = 4
120 ROLL 5, pixels ; 0,175;32,88
130 ROLL 6, pixels ; 0,125;16;176
140 ROLL 7, pixels;; 0,87;32,88
150 ROLL 8, pixels ; 128,175;16,176
160 GOTO 120
```

sau :

```
200 FOR N=1 TO 7;LIST:NEXT N
210 FOR L=1 TO 176;ROLL:50,175,3; NEXT L
```

2.30. SCROLL

- sintaxa : (direction)(,pixels)(; X,Y;hith,lenth)
- tasta : (S)

SCROLL are o sintaxă similară cu ROLL; o diferenţă este că SCROLL poate fi folosit şi fără argumente, caz în care întregul ecran se deplasează în sus cu o linie; SCROLL deplasează întregul ecran în sus cu un pixel; tot ceea ce este împins în afara ecranului este distrus; şi SCROLL este o comandă extrem de utilă pentru jocuri.

2.31. SORT

- sintaxa : SORT string
  SORT string array
  SORT numeric array
- tasta : (M)

SORT ordonează string-uri, numere sau litere, în ordine crescătoare sau descrescătoare.

Exemplu de sortare a unei matrici bidimensionale :

SORT B(1 TO 20) (2)

2.32. SPLIT

- sintaxa : nu este cuvînt cheie, trebuie introdus "()"
- tasta : (simbol shift) + (W) (fără modul grafic)

Dacă o linie pe care o edităm va fi introdusă cu "()" ca prim caracter îm orice instrucţie, partea din linie de dinaintea lui "()" va fi introdusă în program, iar restul va rămîne în zona de editare din partea de jos a ecranului; "()" va dispare şi va fi înlocuit printr-o copie a etichetei originale a liniei; cursorul va fi poziţionat la dreapta acesteia, pentru a se putea modifica cu uşurinţă.

2.33. TRACE

- sintaxa : TRACE număr linie
- tasta : (T)

TRACE este o facilitate care permite depanarea programelor BASIC ; ea are ca efect lansarea cu GOSUB a subrutinei aflate la eticheta specificată în sintaxă.

2.34. UNTIL

- sintaxa : LOOP UNTIL condiţia
  DO UNTIL condiţia
- tasta : (K)

UNTIL are rolul de a permite execuţia condiţionată a instrucţiunilor DO şi LOOP.

2.35. USING

- sintaxa : PRINT USING format string ; număr
- tasta : (U)

Atît comanda USING cît şi funcţia USING$ permit specificarea formatelor pentru numerele care urmează a fi tipărite. Exemplu : moduri de specificare a formatului pentru numărul 12.345 :

```
"11.1"    = 12.3
"111.1"   = 12.3
"000.00"  = 012.35
"$00.00"  = $12.35
```

Funcţia USING$ se utilizează cam în acelaşi fel, cu diferenţa că USING$ poate apărea în orice context, nu numai după PRINT :

PRINT USING A$;număr = PRINT USING$(A$,număr)

2.36. WHILE

- sintaxa : DO WHILE condiţia
  LOOP WHILE condiţia
- tasta : (J)

WHILE se foloseşte în cadrul instrucţiunilor DO şi LOOP pentru a specifica execuţia condiţionată a acestora.

2.37. XOS, XRG, YOS, YRG

Nu sînt cuvinte cheie ci variabile speciale care permit schimbarea scării de reprezentare şi a originii axelor în cadrul execuţiei instrucţiunilor PLOT, DRAW, CIRCLE, FILL.

- XOS, YOS   originea axelor
- XRG, YRG   dimensiunea axelor

Ele nu sînt anulate la RUN sau CLEAR ci sînt poziţionate pe valorile iniţiale (0,0,256,176).

3. FUNCŢII BETA BASIC

3.1. AND

- sintaxa : FN A(număr,număr)
- realizează un AND bit cu bit între două numere care sînt între 0 şi 65535

3.2. BIN$

- sintaxa : FN B$(număr)
- furnizează echivalentul binar al numărului ca un şir de 8 caractere dacă numărul este mai mic de 256 şi un şir de 16 caractere dacă numărul e mai mic de 65535 dar mai mare ca 256.

3.3. CHAR$

- sintaxa : FN C$(număr)
- funcţia converteşte întregi pînă la 65535 într-un string de 2 caractere, permiţînd memorarea unui volum mare de date cu economie de memorie. Echivalentul BASIC al funcţiei este :

```
LET A = INT(număr/256) : LET B = număr - A/256
LET C$ = CHR$ A  +  CHR$ B
```

Dacă încercăm să tipărim string-ul rezultat, obţinem eroarea K (inv.col).

3.4. COSE

- sintaxa : FN C(număr)
- calculează cosinusul numărului de 10 ori mai rapid, dar cu precizie mai mică (4 cifre semnificative).

3.5. DEC

- sintaxa : FN D(string)
- permite conversia unui string de 2 sau 4 caractere care au semnificaţia unui număr în codificarea hexazecimală, în numărul zecimal corespunzător.

3.6. DPEEK

- sintaxa : FN P(adresă)
- este un PEEK pe 2 octeţi de la adresa specificată şi adresa următoare, echivalent cu LET val = PEEK(adr) + 256& PEEK(adr+1)

3.7. FILLED

- sintaxa : FN F()
- furnizează numărul de pixeli umpluţi de ultima comandă FILL.

3.8. HEX$

- sintaxa : FN H$(număr)
- argumentul numeric este convertit într-un string hexazecimal de două caractere dacă numărul este între -255 şi 256 şi de 4 caractere dacă numărul este între -65534 şi 65535.

3.9. INSTRING

- sintaxa : FN I(start,string1,string2)
- caută în stringul 1 după stringul 2 începînd de la caracterul cu numărul de ordine din start.
- dacă îl găseşte, rezultatul va fi poziţia primului caracter din stringul 2 în cadrul stringului 1 ; în caz contrar rezultatul e 0.

Stringul 1 poate fi oricît de lung dar stringul 2 trebuie să fie mai mic de 256 caractere, altfel apare mesajul "invalid argument".

Dacă start=0, apare "subscript wrong".

În stringul 2 poate apare caracterul ";" cu semnificaţia "don't care"

3.10. MEM

- sintaxa : FN M()
- furnizează numărul de octeţi de memorie disponibili.

3.11. MEMORY$

- sintaxa : FN M$()
- furnizează conţinutul memoriei sub formă de caractere (de la 1 la 65535). Exemplu :

```
LET A$ = MEMORY$( 16284 TO 22568 )
```

3.12. MOD

- sintaxa : FN V(număr1,număr2)
- furnizează număr1 modulo număr2

3.13. NUMBER

- sintaxa : FN N(string)
- converteşte un string de 2 caractere într-un întreg (între 1 şi65535)
- echivalent BASIC : LET număr = 256&CODE C$(1) + CODE C$(2)
- se obţine "invalid argument" dacă stringul nu are 2 caractere.
- împreună cu CHER$, NUMBER permite implementarea simplă a matricilor de întregi.

3.14. OR

- sintaxa : FN O(număr,număr)
- realizează un SAU logic bit cu bit între 2 numere.

3.15. RNDM

- sintaxa : FN R(număr)
- dacă numărul este 0, RNDM furnizează un număr aleator între 0 şi 1 (ca şi RND dar este de 2 ori mai rapid), dacă numărul este mai mare furnizează numere aleatoare între 0 şi numărul dat.

3.16. SCRN$

- sintaxa : SCRN$(linie,colqană) la fel ca şi SCREEN$, singura diferenţă este că recunoaşte şi caracterele grafice ale utilizatorului.

3.17. SINE

- sintaxa : FN S(număr)
- furnizează sinusul numărului (este de 6 ori mai rapid, dar mai puţin precis - doar 4 cifre semnificative).

3.18. STRING$

- sintaxa : FN S$(număr,string)
- tipăreşte stringul de "număr" de ori.

3.20. TIME$

- sintaxa : FN T$()
- furnizează timpul curent ca un string de 8 caractere.

Exemplu : LET A$ = TIME$() : PRINT A$

3.21. USING$

- sintaxa : FNUU$("format", număr)
- vezi USING

3.22. XOR

- sintaxa : FN X(număr1,număr2)
- realizează un SAU-EXCLUSIV bit cu bit între 2 numere cuprinse între 1 şi 65535.

ANEXA 1 - Setul de comenzi BETA BASIC

```
ALTER (atribute),variabile
AUTO (linia de început,pasul)
BREAK(îmbunătăţit): Shift + Space
CLOCK număr sau şir de caractere
  Tastele de cursor ca pe SINCLAIR = CAP SHIFT + 5,6,7,8
DEF KEY caracter:instrucţie(:)    sau
        caracter:string
DEF PROC nume procedură
DELETE (prima linie) TO (ultima linie)
DO sau DO WHILE sau DO UNTIL
DPOKE adresa,număr(0-65535)
EDIT(număr linie)
ELSE instrucţie
END PROC
EXIT IF condiţie
FILL (culori INK sau PAPER) x,y
GET variabilă numerică sau variabilă string
JOIN număr de linie
KEY IN string
KEYWORDS 1 sau 0
LIST (număr de linie1) TO (număr de linie2)
LLIST (    -"-        ) TO (      -"-        )
LOOP sau LOOP WHILE sau LOOP UNTIL
ON n GO TO ON număr linie 1, număr linie 2, ...
  sau GOSUB număr linie 1, număr linie 2, ...
ON ERROR număr linie
PLOT x,y;string
POKE adresă string
POP variabilă numerică
PROC nume procedură
RENUM (start TO finish)(LINE new start)(,step)
ROLL direction(pixels)(;x,y width,length)
SCROLL   -"-      -"-     -"-      -"-
SORT string
     sau matrice de caractere (slicer)(slicer)
SPLIT se introduce ca şi () (simbol shift)+(w)
TRACE număr de linie
WHILE
XOS în LET XOS = număr
XRG     -"-     -"-
YOS     -"-     -"-
YRG     -"-     -"-
```

FN A	AND(număr,număr)	FN B$	BIN$(număr)
FN C$	CHR$(număr)	FN C	COSE(număr
FN D	DEC(hex$)	FN P	DPEEK(adresa)
FN F	FILLED()	FNH$	HEX$(număr)
FN I	INSTRING(start,a$,b$)	FN M	MEM()
FN M$	MEMORY$	FN V	MOD(număr,număr)
FN N	NUMBER(string de 2 caractere)	FN O	OR(număr,număr)
FN R	RNDM(număr)	FN K$	SCRN$(linie,coloană)
FN S	SINE(număr)	FN S$	STRING$(număr,string)
FN T$	TIME$()	FN U$	USING(format$,număr)
FN X	XOR(număr,număr)		

ANEXA 2 - Variabile de sistem

Sînt diferite de cele obişnuite. Se dau mai jos :

A.2.1. Variabile grafice

XOS, XRG, YOS, YRG (ANGLE momentan neutilizat) sînt variabile care există tot timpul şi valorile lor afectează execuţia instrucţiunilor PLOT, DRAW, CIRCLE, FILL.

După RUN şi CLEAR ele se poziţionează la valorile standard :
XOS = 0 ; XRG = 256 ; YOS = 0 ; YRG = 176 (ANGLE = 0) .

A.2.2. Variabile pentru TRACE şi ON ERROR

Aceste variabile nu există în mod normal, ele apar doar în cazul utilizării directivei ON ERROR - dacă apare eroare - în intervalul cît aceasta este efectivă, sau în cazul utilizării lui TRACE înainte de fiecare instrucţiune executată atunci cînd TRACE este operaţional.

Nume	Semnificaţie
ERROR	Valoarea codului pentru eroarea apărută
LINE	pentru TRACE numărul liniei care urmează să fie executată . pentru ON ERROR linia la care a apărut ultima eroare
STAT	Furnizează numărul instrucţiei unde a apărut eroarea în cadrul liniei.

Codurile comenzilor BETA BASIC

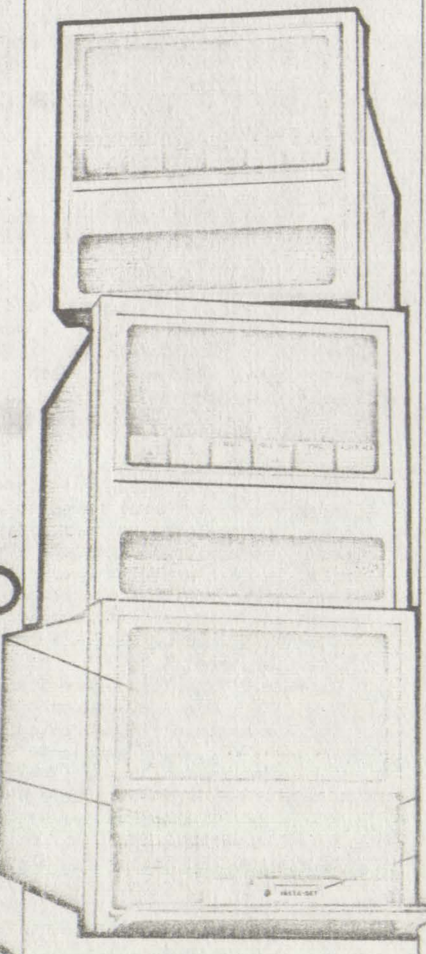
128	8	KEYWORDS	129	1	DEF PROC
130	2	PROC	131	3	END PROC
132	4	RENUM	133		
134	6	AUTO	135	7	DELETE
136	SH+7		137	SH+6	JOIN
138	SH+5	EDIT	139	SH+4	KEY IN
140	SH+1	DEF KEY	144	A	ALTER
145	B		146	C	CLOCK
163	T	TRACE	164	U	USING

ANEXA 3 - Erori specifice BETA BASIC

COD	SEMNIFICAȚIE	CONTEXT
G	No room for line Noul număr al linie coincide cu numărul unei linii care nu este renumerotată sau e mai mare ca 9999	RENUM
S	Missing LOOP EXIT IF sau DO condițional urmat de WHILE sau UNTIL a încercat să ajungă la sfîrșitul unei structuri DO-LOOP și nu a găsit instrucția LOOP	DO,EXIT IF
T	LOOP without DO LOOP fără instrucțiune DO	LOOP
U	No such line DELETE utilizat la un număr de linie inexistent	DELETE
V	No POP data Încercare de citire din stiva GOSUB care la momentul respectiv a fost goală.	POP
W	Missing DEF PROC S-a folosit un PROC și identificatorul respectiv nu a apărut într-o declarație de tip DEF PROC, sau s-a găsit END PROC la un DEF PROC invalid.	PROC,END PROC
X	No END PROC Două DEF PROC-uri succesive, fără END PROC între ele.	DEF PROC
Y	Too hard	RENUM



CALCULATORUL ÎN SPRIJINUL DUMNEAVOASTRĂ



S.L. ING. TORCZKAY TEA

Limbajul de programare micro-PROLOG pentru calculatorul TIM-S

Materialul de fata se refera la implementarea limbajului PROLOG pe microcalculatoarele construite cu microprocesor Z80. Aceasta implementare a fost realizata de firma Logic Programming Associates Ltd. Micro-PROLOG-ul este un dialect al limbajului de referinta PROLOG, propus pentru prima data in perioada anilor 1970.

PROLOG-ul se incadreaza in curentul programarii logice, este un limbaj descriptiv care specifica relatii, deosebindu-se prin acasta radical de limbajele procedurale. In PROLOG dispare diferenta dintre date si proceduri, acestea fiind gestionate in comun ca elemente ale unei baze de date dinamice in memorie. Aceasta baza de date reprezinta de fapt o **baza de cunostinte** (intrucit contine atit date cit si reguli). Extinzind aceasta baza de date pe suport magnetic, capacitatea bazei de cunostinte devine practic nelimitata. Prin aceasta, cit si prin usurinta cu care limbajul permite implementarea prelucrarilor simbolice, PROLOG-ul se preteaza la realizarea unor sisteme bazate pe cunostinte, specifice domeniilor de inteligenta artificiala.

SUMAR :

1.Notiuni despre limbajul micro-PROLOG.

1.1.Termeni

Elementele de baza ale limbajului sint termenii, care se refera la urmatoarele domenii:

1.1.1.Constante

O constanta (simbol) este o succesiune de maximum 60 de caractere, scrise sau nu intre ghilimele.

In componenta constantelor se admit:

a)litere mari
b)litere mici
c)- (caracterul minus)
d)_ (caracterul de subliniere)

Orice alt caracter este separator de constante. Caracterele separatoare de constante pot fi incluse in acestea scriindu-le intre ghilimele.Caracterul ghilimele se scrie intr-o constanta precedat de caracterul "a rond comercial" :@. Constantele care incep cu una din literele X,x,Y,y,Z,z trebuie scrise intre ghilimele (vezi).

Ex.: AB, "Cd", "A1" "B", INST_@"IPTVT@" ,

1.1.2.Numere reale

Numerele reale sint reprezentate in virgula mobila cu exponentul exprimat pe un octet (cu valoarea cuprinsa deci intre -128,-127). Ele se scriu sub forma :+mantisa E+exp. Daca mantisa este scrisa cu punct zecimal, atunci in fata acesteia trebuie sa existe cel putin o cifra. Exponentul poate lipsi.

Ex.: -20.1E-1

1.1.3.Numere intregi cu sau fara semn

Ex.: 10, -20

1.1.4.Liste

Listele sint secvente de termeni scrise intre paranteze mici.

Ex.: (EXEMPLUL 1), (1 2 3), (A B (C D) E)

Un element intr-o lista se poate referi definind "capul" (head) listei si "coada" (tail) listei. Caracterul | (bara verticala) separa capul listei de restul ei.

Ex.: (1|2 3)

1.1.5.Variabile

Constantele care incep cu una din literele X,x,Y,y,Z,z urmate sau nu de maximum 2 cifre reprezinta nume de variabile. Variabilele nu se atribuie. Ele pot fii "libere" (free) sau "legate" (bound) de un termen.[3]

1.2.Predicate logice

1.2.1.Atomi

In limbajul PROLOG se evalueaza **predicate logice** care pot avea valoarea adevarat sau fals. Predicatele logice se identifica (apeleaza) cu ajutorul atomilor. Un atom este o lista in care primul termen este o constanta, care reprezinta numele predicatului definit, urmata sau nu de alti termeni, care reprezinta argumentele predicatului. Se admit si predicate fara argument.

Ex.: (a 1) identifica predicatul cu numele a avind argumentul 1
(a b) identifica predicatul cu numele a avind argumentul b
(TRIUNGHI -1 5 -1) identifica predicatul cu numele TRIUNGHI avind argumentele: -1,5,-1

Intr-un program se pot folosi **predicate definite de utilizator** sau **predicate standard** (predefinite). Predicatele predefinite sint functii implementate prin interpreterul micro-PROLOG, si care se apeleaza conform specificatiilor din paragraful 1.4.

1.2.2.Clauze

Predicatele se definesc cu ajutorul clauzelor, acestea fiind liste de atomi. Primul atom din lista reprezinta atomul de identificare al predicatului. Restul listei reprezinta enumerarea predicatelor care definesc predicatul in cauza. Un predicat este adevarat numai atunci cind toate componentele sale, evaluate in ordinea enumerarii lor, sint adevarate.

Clauzele se pot redacta sub forma declaratiilor ("facts") sau a regulilor ("rules").

Declaratiile sint clauze formate dintr-o lista de atomi, avind un singur termen.

Ex.:((a 1)) ((TRIUNGHI -1 5 1))

Regulile sint clauze formate dintr-o lista de atomi.

Ex.:((VERIFICA)(a 1)(a b))
((CAUTA X)(a x))
((VERTRI Y)(CAUTA Y)(TRIUNGHI -1 5 Y))

Un predicat poate avea mai multe alternative de definire. Toate clauzele referitoare la acelasi predicat se scriu grupate. **Clauzele dintr-un program formeaza baza de date a programului.**

Numarul maxim de variabile admis intr-o clauza este 64.

1.3.Termeni predefiniti

"CON:" - nume fisier consola permanent deschis pentru READ
"LST:" - nume fisier imprimanta permanent deschis pentru WRITE
ALL - folosit cu predicatele SYS,LIST,KILL
& - primul argument al predicatului DICT

1.4.Predicate predefinite si functiile lor

(ABORT)
(ADDCL<clauza>)
(ADDCL<clauza><numar>)
(BORDER<numar>)
(BP<numar><numar>)
(CHAROF<caracter><numar>)
(CL<prototip de clauza>)
(CL<prototip de clauza><numar><variabila>)
(CLMOD<orice>)
(CLOSE<nume de fisier>)
(CLS<numar>)
(CON<termen>)
(CREATE<nume de fisier>)
(CRMOD<nume modul><lista export><lista import>)
(DELCL<prototip de clauza>)
(DELCL<prototip de clauza><numar>)
(DICT)
(EQ<termen><termen>)
(FORALL(<secv.de atomi>)(<secventa atomi>))
(FAIL)



(HYBRID<orice>)
(IF<atom>(<secv.de atomi>)(secv.atomi>))
(INKEY<variabila>)
(INT<numar>)
(INT<numar><numar>)
(INTOK<nume fisier><variabila>)
(ISALL<variabila><termen><secv.de atomi>)
(KILL ALL)
(KILL<constanta>)
(KILL<lista>)
(LESS<numar><numar>)
(LIST ALL)
(LIST<lista>)
(LIST<nume predicat>)
(LISTP<nume fisier>)
(LISTP<nume de fisier><lista>)
(LISTP<nume de fisier><nume de predicat>)
(LNE<numar><numar><numar><numar>)
(LOAD<nume de fisier>)
(LST<termen>)
(NEW<orice>)
(NORMAL<orice>)
(NOT<predicat><secventa de argumente>)
(NUM<termen>)
(OPEN<nume fisier>)
(OPMOD<nume de modul>)
(OR(<secv.de atomi>)(<secv.de atomi>)
(P<secventa de termeni>)
(PIO<numar><numar>)
(PIO<numar><variabila>)
(PNT<numar><numar>)
(PP<secventa de termeni>)
(QT<orice>)
(R<variabila>)
(READ<nume fisier><variabila>)
(RFILL<lista><variabila>)
(RND<numar><variabila>)
(SAVE<nume de fisier>)
(SAVE<nume de fisier><lista>)
(SPACE<variabila>)
(STRINGOF<lista><constanta>)
(SUM<numar><numar><numar>)
(SYS<constanta>)
(TIMES<numar>><numar><numar>)
(VAR<termen>)
(W<nume fisier><lista>)
(WRITE<nume fisier><lista>)
(!<predicat><secventa de argumente>)
(/)
(/*<secventa de termeni>)
(?<predicat><secventa de termeni>)

Functii realizate prin evaluarea predicatelor predefinite

Prin evaluarea unui predicat se pot obtine efecte diferite in functie de natura argumentelor sale. Daca in lista de argumente apar variabile, atunci:

-daca ele sint nelegate, le vom nota cu o, si in cazul evaluarii cu succes a predicatului ele vor fi legate de un termen;
-daca ele sint legate, le vom nota cu i, si la evaluarea clauzei se iau in considerare termenii legati de aceste variabile.Unde este cazul se vor indica posibilitatile de utilizare ("flow patterns").

(ABORT) produce oprirea fortata a programului

(ADDCL<clauza>)
Este intotdeauna "adevarat", adauga clauza in modulul curent sau spatiul de lucru, la sfirsitul clauzelor referitoare la acelasi predicat. Daca predicatul este nedefinit, el se adauga la inceputul bazei de date.

(ADDCL<clauza><numar>)
Adauga clauza in modulul curent sau spatiul de lucru, dupa clauza cu numarul de ordine egal cu <numar>, numar care este relativ fata de inceputul grupului clauzelor pentru acelasi predicat. Validarea predicatului intoarce valoarea "fals" daca <numar>=0 sau daca nu exista clauza cu acest numar.

(BORDER<numar>)
Predicatul intoarce intotdeauna valoarea adevarat, coloreaza border-ul ecranului cu culoarea indicata prin <numar>.

(BP<numar1><numar2>)
Intoarce intotdeauna valoarea "adevarat",produce un sunet de durata egala cu <numar1> si intensitate egala cu <numar2>.

(CHAROF<caracter><numar>)
(ii) (io) (oi) Asociaza sau verifica daca <numar> reprezinta codul ASCII al caracterului mentionat.

(CL<prototip de clauza>)
Intoatce valoarea adevarat daca exista o clauza care se potriveste cu parametrii indicati in prototip. Clauza cautata trebuie sa existe in spatiul de lucru, sau in modulul curent deschis sau sa fie exportata de un modul.

(CL<prototip de clauza><variabila>)
Intoarce valoarea "adevarat" daca exista in baza de date o clauza care se potriveste cu prototipul indicat si numarul de ordine al clauzei in baza de date este mai mare decit <numar>.In cazul predicatului CL, acest numar este numarul de ordine absolut in baza de date.

(CLMOD<orice>)
Intoarce valoarea adevarat daca exista un modul deschis si are ca efect revenirea in spatiul de lucru.Ca argument poate fi folosit orice caracter. Predicatul poate fi folosit ca atom de sine statator, adica fara paranteze.

(CLOSE<nume de fisier>)
Intoarce valoarea "adevarat" daca in momentul apelarii exista un fisier cu numele indicat. In cazul unui fisier deschis pentru READ, se abandoneaza informatia care a ramas necitita in buffer-ul de 256 de octeti, al fisierului. In cazul unui fisier desechis pentru WRITE, se goleste buffer-ul pe banda, si se pune "EOF".

(CLS<numar>)
Intoarce intotdeauna valoarea "adevarat".Are ca efect stergerea ecranului, initializarea fondului (PAPER) pe culoarea indicata de <numar> si aduce cursorul in coltul din stinga sus al ecranului.

(CON<termen>)

Intoarce valoarea "adevarat" daca termenul specificat este o constanta.

(CREATE<nume de fisier>)

Intoarce intotdeauna valoarea "adevarat" si are ca efect deschiderea pe banda a unui fisier pentru WRITE.La un moment dat poat^ fi deschis pe banda un singur fisier. Numele de fisier poate fi orice constanta, formata din maximum 8 caractere, care nu coincide cu termenii predefiniti folositi pentru identificarea fisierelor sistem.

(CRMOD<nume modul><lista export><lista import>)

Intoarce intotdeauna valoarea "adevarat". Are ca efect producerea unui modul, transferind utilizatorul din spatiul de lucru in modulul nou creat (deci se pot introduce definitiile predicatelor in acest modul).La un moment dat poate sa existe un singur modul in afara spatiului de lucru."Lista export" reprezinta declaratii pentru definitiile externe si permite folosirea in spatiul de lucru, a numelor de predicate specificate in modul. "Lista import" reprezinta declaratii pentru referintele externe folosite in modul, si permite deci folosirea in cadrul sau a termenilor si numelor de predicate specificate in spatiul de lucru.

(DELCL<prototip de clauza>)

Intoarce valoarea adevarat daca exista cel putin o clauza care se potriveste cu prototipul si are ca efect stergerea primei clauze de acest tip.

(DELCL<nume de predicat><numar>)

Intoarce valoarea "adevarat" dupa ce sterge clauza (dintre cele care definesc predicatul) cu numarul de ordine specificat.Numarul de ordine se refera la clauzele aceluiasi predicat, deci este un numar de ordine relativ.

(DICT)

Intoarce intotdeauna valoarea "adevarat". Are ca efect listarea urmatoarelor informatii:

a)& - pt. spatiul de lucru
nume modul - daca exista un modul deschis;
b)lista export - daca exista un modul deschis
c)lista import - daca exista un modul deschis
d)constantele - care se folosesc in spatiul de lucru sau modul.

(EQ<termen><termen>)

(ii) - verifica egalitatea celor 2 term.
(io) - leaga (instanteaza) termenul de tip o de valoarea termenului de tip i.
(oi) - -- " --

(FORALL(<secv.de atomi>)(<secv.de atomi>))

Intoarce valoarea "adevarat" daca si numai daca pentru toate situatiile posibile cind prima secventa este valida, a doua secventa este valida.Acest predicat corespunde implicatiei logice.

(FAIL)

Intoarce intotdeauna valoarea "fals". Poate fi folosit si ca atom de sine statator.

(HYBRID<orice>)

Intoarce intotdeauna "adevarat" si comuta spatiul de lucru pe ecran in "lower screen", eliberind "upper screen" pentru grafica.Prin definitie, dintre cele 24 de linii ale ecranului, primele 22 reprezinta "upper screen", iar ultimele 4, "lower screen".In grafica trebuie sa se tina cont de suprapunerea celor doua linii din definitia celor doua ecrane.

(IF<atom>(<secv.1 atomi>)(<secv.2 atomi>))

Efectul este urmatorul:

a)daca atomul este adevarat, atunci:
-intoarce "adevarat" daca secv.1 de atomi este adevarata;
-intoarce "fals" daca secv.1 de atomi este falsa.

b)daca atomul este fals, atunci:
-intoarce "adevarat" daca secv.2 de atomi este adevarata;
-intoarce "fals" daca secv.2 de atomi este falsa.

Pentru ca predicatul IF sa conditioneze validarea nu a unui singur atom, ci a unei secvente de atomi, se foloseste predicatul predefinit ? (semnul intrebarii).

(IF(?(<secv.atomi>))
(<secv.1 atomi>)(<secv.2 atomi>))

(INKEY<variabila>)

(o) Intoarce valoarea "adevarat" legind variabila de caracterul corespunzator tastei apasate. Predicatul nu asteapta apasarea unei taste. Daca utilizatorul nu a tastat nimic, variabila se leaga de caracterul nul, avind codul 255.

(INT<numar>)

(i) Intoarce valoarea "adevarat" daca numarul este intreg.

(INT<numar 1><numar 2>)

(ii) Intoarce valoarea "adevarat" daca numar 2 este partea intreaga a lui numar 1.
(io) Intoarce valoarea "adevarat" legind numar 2 (variabila nelegata) de partea intreaga a lui numar 1.

(INTOK<nume fisier><variabila>)

Intoarce valoarea "adevarat", citeste o constanta dintr-un fisier deschis pentru READ si leaga variabila de aceasta constanta. Intoarce valoarea "fals" la EOF.Predicatul INTOK interpreteaza toate caracterele drept constante, deci spre deosebire de READ, tine cont de separatorii de constanta.

(ISALL<variabila><termen><secv.atomi>)

Construieste lista tuturor termenilor de forma <termen> pentru toate situatiile cind <secv.atomi> poate fi validata. Variabila este legata cu lista astfel obtinuta si intoarce valoarea "adevarat".

(KILL<constanta>)

Intoarce valoarea "adevarat" daca <constanta> (diferita de o constanta predefinita), reprezinta un nume de predicat din spatiul de lucru sau modulul curent deschis. Are ca efect stergerea tuturor clauzelor referitoare la acest predicat. Intoarce "fals" in cazul in care constanta este predefinita.

(KILL<lista constante>)

Are efectul predicatului definit mai sus, pentru fiecare constanta din lista specificata.Daca o constanta indica un nume de predicat pentru care nu exista nici o clauza in baza de date, se intoarce valoarea "adevarat" si se trece la constanta urmatoare.

(KILL ALL)

Intoarce valoarea "adevarat" si sterge toate clauzele din baza de date a spatiului de lucru si modulul curent deschis.

(LESS<numar 1><numar 2>)

(ii) Intoarce valoarea "adevarat" daca <numar 1> mai mic decit <numar 2>.

(LIST<nume de predicat>)
Intoarce "adevarat" si listeaza la consola toate clauzele referitoare la numele de predicat, daca acesta se gaseste in spatiul de lucru sau este exportat de modulul curent deschis. Intoarce valoarea "fals" daca <nume de predicat> este o constanta predefinita.

(LIST<lista>)
Are efectul predicatului LIST definit anterior,pentru fiecare element specificat in lista.

(LIST ALL)
Intoarce intotdeauna valoarea "adevarat" si are efectul listarii la consola a intregii baze de date.
Listarea poate fi intrerupta apasind simultan tastele Symbol Shift si A.
Listarea se poate relua tastind ENTER.

(LISTP<nume fisier><nume predicat>)
Intoarce valoarea "adevarat" si daca <nume de predicat> exista in spatiul de lucru curent sau este exportat de modulul curent deschis, atunci are ca efect listarea tuturor clauzelor referitoare la acest predicat, in fisierul specificat. Fisierul poate sa fie pe banda, deschis cu CREATE, sau poate sa fie un fisier predefinit deschis pentru WRITE, adica CON:,LST:.
Intoarce valoarea "fals" daca se cere listarea unui predicat sau a unei constante predefinite.

(LISTP<nume fisier><lista predicate>)
Are efectul predicatului LISTP definit mai sus, pentru fiecare nume de predicat din lista specificata.

(LISTP<nume fisier>)
Intoarce intotdeauna valoarea "adevarat" si listeaza in fisier (banda sau perifericul cerut) toate clauzele din baza de date a spatiului de lucru sau a modulului curent deschis.

(LNE<numar 1x><numar 1y><numar 2x><numar 2y>)
(iiii) Intoarce valoarea "adevarat" si are ca efect trasarea unei linii pe ecran intre punctele specificate de cele doua perechi de coordonate.
In modul grafic (inalta rezolutie), ecranul este definit ca o matrice de 256x176 pixeli, punctul de coordonate 0,0 aflindu-se in centrul ecranului. Deci pe axa absciselor, valorile pot varia intre [-128,127], iar pe axa ordonatelor intre [-88,87]. Pentru a evita suprapunerea imaginilor realizate in zona "upper screen" cu textele ce apar in zona "lower screen", se recomanda ca valorile pentru ordonate sa varieze intre [-80,87].

(LOAD<nume fisier>)
Intoarce valoarea "adevarat" daca gaseste pe banda fisierul cu numele specificat, fisier care a fost creat prin SAVE.

(LST<termen>)
(i) Intoarce valoarea "adevarat" daca termenul specificat este o lista.

(NEW<orice>)
Intoarce valoarea "adevarat" si elibereaza complet spatiul de lucru.Nu initializeaza atributele si pozitia cursorului.

(NORMALL<orice>)
Intoarce valoarea "adevarat" si are ca efect revenirea in modul de lucru "upper screen". Are efect invers cu predicatul HYBRID.

(NOT<predicat><secv.argumente>)
Intoarce valoarea "adevarat" daca atomul (<predicat><secv.argumente>) este "fals".Pentru a produce evaluarea unei secvente de atomi, se foloseste predicatul ? (semnul intrebarii).

(NOT ?(secv. atomi)))

(NUM<termen>)
(i) Intoarce valoarea "adevarat" daca <termen> este un numar.

(OPEN<nume fisier>)
Intoarce valoarea "adevarat" daca intilneste pe banda un fisier cu numele specificat format din maximum 8 caractere. Are ca efect deschiderea fisierului pentru READ. Daca <nume fisier> este constanta vida "" (ceea ce la CREATE nu se permite insa),atunci prin OPEN se deschide primul fisier intilnit pe banda.

(OPMOD<nume modul>)
Intoarce valoare "adevarat" daca modulul exista deja creat in memorie. Are ca efect alternarea spatiului de lucru cu modulul deja deschis.In felul acesta, clauzele din modul devin accesibile pentru editare.

(OR(<secv.1 atomi><secv.2 atomi>)
Intoarce valoarea "adevarat" daca fie <secv.1 atomi> fie <secv.2 atomi> este adevarata.

(P<secventa termeni>)
Intoarce intotdeauna valoarea "adevarat" si are ca efect tiparirea pe ecran a termenilor, fara a imprima ghilimele, fara <CR> si tinind cont de caracterele de control. La terminarea tiparirii, nu lasa nici un caracter spatiu si ramine pe aceeasi linie.

(PP<secventa termeni>)
Intoarce intotdeauna valoarea "adevarat" si are ca efect tiparirea pe ecran a <secventei de termeni> cu <CR> si imprima ghilimele in jurul separatorilor.

(PIO<numar 1><numar 2>)
(ii) - Intoarce valoarea "adevarat" si are ca efect inscrierea valorii reprezentata prin <numar 2> pe portul <numar1>
(oi) - Intoarce valoarea "adevarat" si leaga variabila (termenul o) de valoarea de pe portul (numar 1).

(PNT<numar 1><numar 2>)
Intoarce valoarea "adevarat" si are ca efect inscrierea pe ecran a unui punct de coordonate <numar 1>,<numar 2>.

(QT<orice>)
Intoarce intotdeauna valoarea "adevarat" si are ca efect revenirea la sistemul de operare. Predicatul este accesibil numai in microcalculatoarele functionind sub sistemul de operare CP/M.

(R<variabila>)
Intoarce valoare "adevarat" si are ca efect citirea de la consola a primului termen pe care il gaseste in buffer-ul de intrare, fara a-l goli.

(READ<nume fisier><variabila>)
Intoarce valoarea "adevarat" daca exista deschis cu OPEN fisierul cu numele specificat, si are ca efect citirea unui termen din acest fisier in variabila indicata. Fisierul poate fi si fisierul predefinit "CON:".

(RFILL<lista><variabila>)
Intoarce valoarea "adevarat".Actiunea acestui predicat este complexa:
- tipareste la consola continutul listei luata din baza de date sau de pe banda,lista care reprezinta o clauza selectata cu predicatul CL
- pune cursorul in stinga liniei tiparite pe ecran si permite utilizatorului sa editeze linia
- leaga <variabila> de continutul lis-

[illegible] editate in momentul apasarii pe tasta ENTER.

(RND<numar><variabila>)
(io) Intoarce valoarea "adevarat" si leaga variabila de un numar aleator cuprins intre [0,<numar>].

(SAVE<nume fisier><nume predicat>)
Intoarce valoarea "adevarat" si are ca efect deschiderea pe banda a unui fisier cu numele specificat, listarea in el a tuturor clauzelor predicatului specificat si inchiderea fisierului.

(SAVE<nume fisier>(<secventa predicate>))
Are efectul predicatului SAVE mai sus definit pentru fiecare predicat din secventa indicata.

(SAVE<nume fisier>)
Are efectul predicatului SAVE mai sus definit, salvind intreaga baza de date pe fisier.

(SPACE<variabila>)
(o) Intoarce valoarea "adevarat" si leaga variabila de un numar care reprezinta numarul de Kocteti liberi in memorie.Se mentioneaza ca la initializarea sistemului, utilizatorul are la dispozitie 24153 octeti.

(STRINGOF<lista><constanta>)
(ii) Intoarce valoarea "adevarat" daca <lista> contine exact caracterele care apar in <constanta>.
(io) Intoarce valoarea "adevarat" dupa ce
(oi) leaga variabila (temenul o) de termenul celalalt (de tip i).

(SUM<numar 1><numar 2><numar 3>)
(iii) Intoarce valoarea "adevarat" daca <numar 3> este suma celorlalte doua.
(iio) Intoarce valoarea "adevarat" dupa ce
(ioi) leaga variabila (termenul o) de suma
(oii) sau diferenta celorlalti doi termeni (de tip i).

(SYS<constanta>)
Intoarce valoarea "adevarat" daca <constanta> este predefinita,de ex.ALL.

(TIMES<numar 1><numar 2><numar 3>)
(iii) Intoarce valoarea "adevarat" daca <numar 3> este produsul celorlalte doua
(iio) Intoarce valoarea "adevarat" dupa ce
(ioi) leaga variabila <termenul o) de
(oii) produsul sau citul celorlalti doi termeni (de tip i).

(VAR<termen>)
Intoarce valoarea "adevarat" daca termenul este o variabila nelegata (neinstantiata).

(W<nume fisier><lista>)
Se comporta exact ca WRITE cu exceptia faptului ca toate caracterele sint interpretate ca atare (caracterele de control functionind drept caractere de control), si in cazul perifericelor nu se trimite <CR> si nici nu se pun ghilimele.

(WRITE<nume fisier><lista>)
Intoarce valoarea "adevarat" daca fisierul este deschis pentru WRITE, si scrie in fisier (pe banda sau periferic) termenii din <lista>. Pentru caracterele de control nu se trimit caractere ASCII, ci se trimit constantele corespunzatoare. Daca perifericul este "CON:" sau "LST:" (consola sau ZX-printer), atunci dupa ce tipareste termenul din lista, genereaza <CR>.
Separatorii sint tipariti intre ghilimele.

(!<predicat><secventa argumente>)
Obliga atomul (<predicat><secventa argumente>) sa aiba o singura solutie.
Intoarce valoarea "fals" daca atomul nu poate fi validat.

(/*<secventa termeni>)
Intoarce intotdeauna valoarea "adevarat" si se foloseste pentru inserarea comentariilor in clauze.

(/) sau / daca este folosit ca atom de sine statator
Intoarce intotdeauna valoarea "adevarat", impiedica BACKTRACKING-ul in stinga clauzei in care se foloseste si descarca in mod corespunzator stiva.

(?(<secventa de atomi>))
Intoarce valoarea "adevarat" daca si numai daca intreaga secventa de atomi poate fi validata. Folosirea acestui predicat este utila atunci cind:
- vrem sa se execute o secventa de atomi fara ca aceasta sa se introduca in baza de date
- dorim sa se insereze intr-o clauza o secventa de atomi, acolo unde din punct de vedere sintactic ar trebui sa existe un singur atom (de exemplu daca se foloseste predicatul NOT pentru o secventa de atomi).

1.5.Setul de caractere

Setul de caractere recunoscut de microPROLOG se poate clasifica in functie de codurile caracterelor in urmatoarele categorii:

a) Caractere de control, avind codurile cuprinse intre 0-31;
b) Caractere ASCII, avind codurile cuprinse intre 32-127.Codul 96 corespunde "lirei sterline", iar 127 simbolului "copyright";
c) Caractere grafice, avind codurile cuprinse intre 128-143;
d) Caractere grafice,definite de utilizator, avind codurile cuprinse intre 144-164;
e) Caractere netiparibile, avind codurile cuprinse intre 165-255.

Apasind o tasta a calculatorului, se produce acelasi cod ASCII ca si in BASIC, dar el poate avea o alta semnificatie.

a)Caracterele de control

Cod	Reprezentare	Obtinere	Semnificatie
0	?		
1	"@A"		
2	"@B"		
3	"@C"		
4	"@D"		
5	"@E"		
6	"@F"		PRINT comma control
7	"@G"	CS+1	
8	"@H"		BACK SPACE
9	"@I"		
10	"@J"	CS+6	
11	"@K"	CS+7	
12	"@L"		DELETE
13	"@M"	CR	carriage return
14	"@N"		
15	"@O"		
16	"@P"	E+CS+1	INK control
17	"@Q"	E+1	PAPER control
18	"@R"		FLASH control
19	"@S"		BRIGHT control
20	"@T"	CS+4	INVERSE control
21	"@U"		
22	"@V"		OVER control
23	"@W"		AT control
24	"@X"		TAB control
25	"@Y"		
26	"@Z"		
27	"@["		
28	"@\"		
29	"@]"		
30	"@^"		
31	"@_"		

Rubrica "reprezentare" se refera la modul in care sint tiparite codurile respective, atunci cind ele sint obiectul unui PP sau RFILL sau (WRITE "CON:").

Rubrica "semnificatie" se refera la modul in care sint interpretate caracterele atunci cind ele apar intr-o constanta tiparita cu P sau (W "CON:").

Prin E s-a notat modul E pentru cursor, prin CS s-a notat tasta CAPS SHIFT si prin SS tasta SYMBOL SHIFT.

b)Caractere tiparibile ASCII

Aceste caractere se obtin in modul cunoscut din BASIC.

c)Caractere grafice

Aceste caractere se obtin in modul cunoscut din BASIC.

d)Caractere grafice definite de utilizator

Se obtin cu cursorul in mod G, definind taste cu functii grafice pentru care trebuie sa existe UDG-urile definite corespunzator. In micro-PROLOG, stiva microprocesorului se defineste la sfirsitul RAM-ului fizic (de la 65535 in jos). UDG-urile trebuie incarcate in memorie inainte de incarcarea interpreterului micro-PROLOG. Memtionam ca prin incarcarea acestuia in memorie, exista pericolul acoperirii UDG-urilor prin stiva.

1.6.Structura unui program micro-PROLOG

1.6.1.Principiile de evaluare ale unui predicat logic

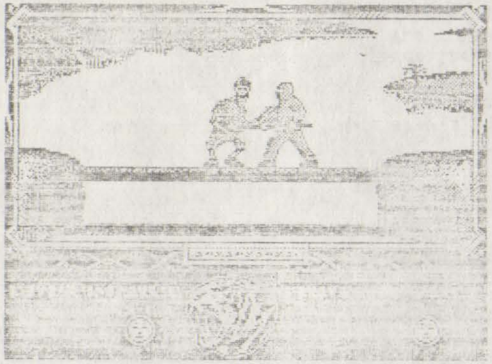
Dupa cum s-a aratat, un program micro-PROLOG este o succesiune de clauze reprezentind declaratii sau reguli pentru diferite predicate logice.Lansarea in executie la un moment dat a programului (vezi 2.3.) inseamna o cerere de evaluare a unui predicat logic.

Un **predicat este valid** (adevarat) daca toate componentele sale (adica toate predicatele care il definesc) sint valide.

Un predicat poate fi evaluat
- daca este definit, adica exista clauze care se refera la el;
- si daca exista cel putin o clauza al carei numar de argumente sa fie egal cu numarul argumentelor predicatului ce urmeaza sa fie validat.

Se mentioneaza ca limbajul micro-PROLOG admite si predicate care pot fi definite prin clauze cu un numar diferit de argumente.

Predicatele definite de limbaj pot fi "adevarate" sau "false" in functie de argumentele care li se transmit (vezi 1.4.).



Predicatele definite de utilizator prin reguli pot fi si ele adevarate sau false, in functie de rezultatul validarii componentelor lor.

Atunci cind un predicat este "fals", sistemul micro-PROLOG <u>incearca</u> <u>reevaluarea</u> folosind principiul BACKTRACKING-ului. Daca nici prin aceasta reevaluare predicatul nu poate fi validat, atunci se tipareste ? (semnul intrebarii) si interpreterul asteapta o noua intrare din partea utilizatorului.

Prin **BACKTRACKING** se intelege o cautare inapoi. Fie de exemplu predicatul Pi definit prin:
- regula Ri1 specificata prin predicatele P2 si P3;
- regula Ri2 specificata prin predicatele P4, P5,...Pm
- regula Rin specificata printr-o succesiune oarecare de predicate.

Daca la incercarea de validare initiala a lui Pi prin Ri1 se obtine valoarea "fals",

1) si predicatele P2 si P3 care determina regula Ri1, au cite o singura definitie, atunci BACKTRACKING-ul inseamna : incercarea de reevaluare a lui Pi1 prin regula Ri2 (presupunind ca in baza de date Ri2 urmeaza imediat dupa Ri1). Daca pentru componentele lui Ri2 se repeta aceleasi conditii ca si cele mai sus aratate, atunci reevaluarea se continua evaluind restul clauzelor de tip Ri. In momentul in care o regula devine adevarata, predicatul Pi devine "adevarat".

2) Si daca fie P2, fie P3 sint predicate definite prin mai multe clauze, atunci BACKTRACKING-ul inseamna: incercarea de reevaluare a lui Pi prin reevaluarea fie a lui P3, fie a lui P2, fie a ambelor (luind din baza de date urmatoarea clauza care defineste unul dintre predicatele P2 sau P3). Daca nici o reevaluare a unuia din aceste componente nu valideaza pe Pi, atunci se trece la o noua reevaluare folosind regula Ri2 de definire a lui Pi s.a.m.d.

La evaluarea unui predicat, sistemul marcheaza in stiva toate punctele posibile de BACKTRACKING din structura predicatului.Aceste informatii sint folosite in cazul in care se incearca reevaluarea predicatului prin BACKTRACKING.Deci in cazul unor predicate incorecte ca mod de conceptie, se ajunge la depasirea stivei, semnalata prin mesajul "no space left" (vezi 2.3.),care provoaca intreruperea fortata a programului.

Imbunatatirea performantelor de programare se poate realiza folosind predicatul predefinit "/" care impiedica BACKTRACKING-ul la stinga intr-o clauza si in consecinta descarca in mod corespunzator stiva.

Viteza de calcul poate fi marita folosind predicatul predefinit "!" care are drept efect selectarea unei singure clauze (prima intilnita in baza de date) in cazul BACKTRACKING-ului.

Pentru imbunatatirea performantelor de programare, se recomanda urmatoarele:
- in cazul predicatelor definite recursiv, este util sa se foloseasca "definitiile recursive prin coada" (tail recursive definitions). Aceasta inseamna ca cel mult o clauza din definitia unui predicat sa faca apel recursiv la predicatul care se defineste si aceasta clauza sa fie ultima din definitia predicatului [1];
- definitiile de forma "tail recursive" sa fie inlocuite daca este cazul prin iteratii [3], ceea ce duce la inlocuirea recursivitatii cu un sir de evaluari de predicate;
- la definirea unui predicat sa se foloseasca eventual mai multe variabile decit predicate, pentru a micsora numarul predicatelor componente [3];
- pentru implementarea ciclurilor de program, se recomanda sa se foloseasca BACKTRACKING-ul in locul apelurilor recursive [3].

1.6.2.Exemplu de program micro-PROLOG format dintr-un singur modul

In fig. 1 se prezinta modul cum am realizat predicatul SORT, care implementeaza algoritmul lui Hoare [4], pentru sortarea unei liste. Sortarea propusa de Hoare consta in impartirea unei liste data (H|T), unde H reprezinta capul listei si T - coada listei, in doua liste L si M astfel incit :

- toate elementele din L sa fie <= cu H
- toate elementele din M sa fie > decit H
- ordinea elementelor in L si M este aceeasi ca si in lista data initial.

Predicatul SORT foloseste predicatele ADAUGA (pentru concatenarea a doua liste) si DIVIDE (pentru crearea unor subliste in functie de o conditie data).

```
((SORT () () )
  /)
((SORT (X|Y) Z)
  (DIVIDE X Y x y)
  (SORT x z)
  (SORT y X1)
  (ADAUGA z (X|X1) Z)
  /)
((QSORT () () )
  /)
((QSORT (X|Y) Z)
  (P "@M@MQSORT:" " X=" X "Y="
Y "Z=" Z)
  (DIVIDE X Y x y)
  (P "@M@MIESIRE DIVIDE  x=" x "
y=" y)
  (QSORT x z)
  (QSORT y X1)
  (P "@M@MINTRARE ADAUGA z=" z
"X=" X "X1=" X1)
  (ADAUGA z (X|X1) Z)
  (P "@M@MIES ADAUGA @MLISTA SOR
TATA:" Z "@M")
  /) 
((ADAUGA (X|Y) Z (X|x))
  (ADAUGA Y Z x))
((ADAUGA () X X)
  /)
((DIVIDE X () () () )
  /)
((DIVIDE X (Y|Z) x (Y|y))
  (NOT LESS Y X)
  /
  (DIVIDE_X Z x y))

((DIVIDE X (Y|Z) (Y|x) y)
  (LESS Y X)
  /
  (DIVIDE X Z x y))

((TEST_SORT 1)
  (EQ X (5 2 8))
  (P "@M@MLISTA NESORTATA " X)
  (SORT X Y)
  (P "@MLISTA SORTATA " Y "@M"))

((TEST_SORT 2)
  (EQ X (C G A))
  (P "@M@MLISTA NESORTATA " X)
  (SORT X Y)
  (P "@MLISTA SORTATA " Y "@M"))

((TEST_SORT 3)
  (EQ X (4 2))
  (QSORT X Y ))

((TEST ADAUGA_1)
  (ADAUGA (1 2 3) (4 5) X)
  (PP LISTA REZULTAT X))

((TEST ADAUGA_2)
  (ADAUGA ((1 2) 3 (4 5)) (6 7)
X)
  (PP LISTA REZULTAT X))

((TEST ADAUGA_3)
  (ADAUGA (((1 2) (A B) (1 2 3))
(10 C) X)
  (PP LISTA REZULTAT X))

((TEST 5)
  (P "@MTEST PT. INKEY@MAPASATI
ORICE TASTA ")
  (CICLU X))

((CICLU X)
  (NOT EQ X "")
  (P "@MATI APASAT PE: " X "@M"))
((CICLU X)
  (INKEY Y)
  (CICLU Y)/)
```

FIG.1



Predicatul QSORT este o varianta a predicatului SORT, in el folosindu-se tipariri de control in scopul urmaririi modului de functionare a interpretorului micro-PROLOG.

Prin comanda TEST_SORT 1 (vezi 2.3) se cere evaluarea predicatului SORT pentru a prelucra o lista numerica cu elemente numerice.

Comanda TEST_SORT 2 lanseaza predicatul SORT pentru sortarea unei liste cu componente de tip alfanumeric.

Comanda TEST_SORT 3 provoaca validarea predicatului QSORT pentru o lista cu doua elemente numerice.

Comenzile TEST ADAUGA_1
TEST ADAUGA_2
TEST ADAUGA_3
ilustreaza folosirea predicatului ADAUGA. Prin ele am urmarit exemplificarea posibilitatii definirii unor **termeni compusi.**

Comanda TEST 5 exemplifica folosirea predicatului INKEY.

1.6.3.Programarea modulara

Conceptul de programare modulara se realizeaza in micro-PROLOG prin definirea "spatiului de lucru" (work space) si a "modulului curent deschis" [1].

La initializarea sistemului, utilizatorul se afla in spatiul de lucru. El are posibilitatea ca la un moment dat sa defineasca si sa foloseasca in afara spatiului de lucru un singur modul de program.

Un modul de program este caracterizat prin:

- numele modulului
- "lista export" a modulului
- "lista import" a modulului

Predicatele sint locale spatiului de lucru sau modulului in care au fost definite. Pentru a putea fi apelate din exteriorul unitatii de program in care au fost definite, ele trebuie sa se specifice drept referinte externe.Aceste referinte externe sint mentionate in lista de export, respectiv import, a unui modul.

Numele modulului este o constanta care trebuie sa fie diferita de numele oricarui predicat.

Lista export reprezinta lista numelor de predicate definite in modul, care pot fi apelate din spatiul de lucru.

Lista import contine numele de predicate si constante definite in spatiul de lucru, care vor fi folosite in clauzele din interiorul modulului. Predicatele definite si exportate de catre un modul pot fi listate in spatiul de lucru.

Modulele se gestioneaza cu ajutorul urmatoarelor predicate predefinite:

- CRMOD - creaza un modul;
- OPMOD - deschide un modul si comuta utilizatorul din spatiul de lucru in modul, permitind editarea clauzelor din acesta;
- CLMOD - inchide modulul;
- KILL - sterge un modul sau mai multe module;
- NEW - sterge toate modulele din memorie.

Pentru salvarea unui modul pe banda, se poate folosi secventa prezentata in continuare [1]:

```
?((OPMOD modul_mod))   -deschide modulul
  (CREATE MODUL)       -deschide fisierul
  (DICT X Y Z!x)       -se obtin parametrii
                       modulului:nume, liste
                       import, export
  (WRITE MODUL(X Y Z)) -scrie caracteristi-
                       cile  modulului  in
                       fisier
  (LISTP  MODUL)       -listeaza  clauzele
                       din modul in fisier
  (WRITE MODUL(CLMOD))  -scrie in fisier
                       CLMOD
  (CLOSE MODUL)         -inchide fisierul
  CLMOD)                -inchide modulul
```

Cu ajutorul predicatului predefinit LOAD se pot incarca de pe banda clauze in spatiul de lucru sau in modulul curent creat. Pentru ca LOAD sa creeze un modul, pe banda trebuie sa existe urmatoarele informatii:

- o constanta care va fi interpretata ca nume de modul
- o lista care va fi interpretata ca lista de export
- o a doua lista care va fi interpretata ca lista de import
- clauze care se incarca de pe banda si se adauga in baza de date a modulului
- constanta CLMOD care determina inchiderea modulului.

2. Editarea programelor

2.1. Tastatura microcalculatorului TIM-S si functiile micro-PROLOG

In micro-PROLOG se utilizeaza urmatoarele moduri de lucru: C,L,E,G. Modul K nu exista, toate numele de predicate (inclusiv cele predefinite) tastindu-se caracter cu caracter.

Cursorul se alterneaza ca in BASIC pentru modurile C,L,E si G.

De la tastatura sint accesibile numai simbolurile cu valoare de caracter, deci tentativa de tiparire a unui simbol compus duce la tiparirea pe ecran a unui "?", sau are drept rezultat introducerea unui caracter de control.

2.2. Introducerea si editarea programelor

Utlizatorul interactioneaza cu micro-PROLOG-ul prin intermediul consolei. O sesiune de lucru decurge astfel:

- se initializeaza microsistemul
- se incarca de pe caseta interpreterul micro-PROLOG, de exemplu LOAD "PROLOG".

Dupa incarcare apare mesajul

24513 Bytes free

urmat de prompterul "&" in coltul din stinga jos al ecranului. In acest moment poate incepe editarea programului.

Utilizatorul este in dialog permanent cu sistemul, care citeste informatia introdusa prin intermediul tastaturii si tipareste pe ecran raspunsurile sistemului.

O linie de program se poate extinde pe mai multe linii ecran, ea fiind preluata de sistem in momentul apasarii pe tasta ENTER. Pentru abandonarea unei linii introduse (adica sa nu fie preluata in baza de date din sistem), inainte de a apasa pe tasta ENTER, se apasa simultan pe SS+SPACE.

Editarea pe orizontala a unei linii program in momentul introducerii ei, se realizeaza cu functii similare ca si in BASIC, si anume: CS+ ⟵, CS+ ⟶, DELETE.

Functiile de **editare pe verticala** (CS+ ↓, CS+ ↑) nu sint accesibile micro-PROLOG-ului. O linie de program odata introdusa si adaugata bazei de date, nu mai este accesibila prin functii primare de editare ale sistemului.

Sistemul micro-PROLOG pune prompterul "&" si apeleaza o functie interna de citire din buffer care pune "." pe ecran. In acest moment se poate introduce o linie de program. De exemplu:

((A 1))

Controlul revine imediat utilizatorului pentru a continua sa tipareasca la consola, in trei situatii:

- daca nu s-a tiparit nimic, apare "."
- daca exista ghilimele deschise care nu au fostinchise, se comporta ca mai sus,ceea ce se testeaza in continuare, facind parte din constanta inceputa anterior, incluzindu-se de asemenea si <CR>
- daca exista paranteze deschise care nu au fost inchise, se tipareste un "numar" care reprezinta numarul de paranteze ramase deschise. Se trece la interpretarea textului numai dupa ce au fost tastate toate parantezele lipsa.

In momentul apasarii pe ENTER, se face o analiza a liniei introduse. Daca ea reprezinta o lista, atunci ea se adauga in baza de date, in grupul predicatelor cu acelasi nume.

Daca linia este o comanda, atunci se cere validarea predicatului specificat prin comanda respectiva (vezi 2.3.). Prin comanda se intelege un nume de predicat care este definit ca avind un singur argument.

Daca se transmite o secventa de forma

?((TEST 1))

atunci aceasta clauza nu se introduce in baza de date, ci se incearca evaluarea ei.

Listarea bazei de date se cere prin comanda LIST ALL. Nu exista notiunea de SCROLL, listarea se intrerupe cu SS+A si se reia cu ENTER.

2.3. Lansarea in executie a programelor

Tastind un nume de predicat care are in definitia sa un singur argument, urmat de o constanta care este interpretata drept acel argument, se realizeaza "lansarea in executie" a unui program, adica tentativa de validare a predicatului indicat.

Daca predicatul este validat, pe ecran apare "&", in caz contrar "?".

De exemplu:

Comanda	Informatie rezultata pe ecran
TEST_SORT 1	LISTA NESORTATA (5 2 8) LISTA SORTATA (2 5 8)
TEST_SORT 2	LISTA NESORTATA (C G A) LISTA SORTATA (A C G)
TEST_SORT 3	QSORT: X= 4 Y= (2) Z= X IESIRE DIVIDE x= (2) y= () QSORT: X= 2 Y= () Z= X IESIRE DIVIDE x= () y= () INTRARE ADAUGA z= () X= 2 X1= () IES ADAUGA LISTA SORTATA: (2) INTRARE ADAUGA z= (2) X= 4 X1= () IES ADAUGA LISTA SORTATA: (2 4)
TEST ADAUGA_1	LISTA REZULTAT (1 2 3 4 5)
TEST ADAUGA_2	LISTA REZULTAT ((1 2) 3 (4 5) 6 7)
TEST ADAUGA_3	LISTA REZULTAT (((1 2) (A B) (1 2 3)) 10 C)
TEST 5	TEST PT. INKEY APASATI ORICE TASTA ATI APASAT PE: s

```
TEST PT. INKEY
APASATI ORICE TASTA No space left
Obs.:S-a asteptat prea mult.
```

2.4.Coduri de eroare [1]

Cod	Semnificatie	Apare la folosirea predic.
0	Overflow	SUM, TIMES
1	Underflow	TIMES
2	No definitons for relation	Apelarea unui predicat nedefinit
3	Too many variables or invalid form	CL, ADDCL, SUM, TIMES, in cazul in care se folosesc mai multe sau mai putine argumente decit cere predicatul predefinit
4	Error in adding clauses	ADDCL
5	File error (Tape loading error)	LOAD, READ
6	Unclosed file (Close last used file first)	LOAD, SAVE, CREATE, OPEN
11	Break	La apasare SS+SPACE
12	Illegal use of modus	OPMOD, CRMOD, CLMOD,
13	Line or point of screen	PNT, LNE
22	Invalid colour	P sau W (la "CON:") cu caractere de control

2.5.Exemplu de program utilitar pentru editare

Pentru a usura munca de editare a programelor, am realizat un sumar editor prin predicatul ED. El implementeaza urmatoarele functii: I,D,M,E. Pentru fiecare functie, trebuie sa se cunoasca numele predicatului prelucrat si numarul clauzei din definitia predicatului asupra caruia actioneaza editorul. (Fig.2)

Functiile sint:

- I inserarea unei clauze
- D stergerea unei clauze
- M. modificarea unei clauze, cu stergerea vechii clauze
- E modificarea unei clauze, cu pastrarea vechii clauze.

In continuare se prezinta programul ED.

Folosirea editorului se poate exemplifica astfel:

- introduceti clauzele ((Q 1))
  ((Q 2))
- folositi functia de inserare I prin:
  tastati ED I
  la intrebarea NUME PREDICAT:
  raspundeti Q
  la intrebarea NUMAR CLAUZA:
  raspundeti 1
  la mesajul SCRIETI CLAUZA:
  raspundeti ((Q I))
  apoi listati predicatul Q prin: LIST Q.

In urma acestor actiuni, pe ecran va apare

```
((Q 1))
((Q I))
((Q 2))
```

Pentru celelalte functii ale editorului, realizati urmatorul dialog:

```
ED E
NUME PREDICAT:
.Q
NUMAR CLAUZA:
.2
((Q E))
LIST Q
((Q 1))
((Q E))
((Q I))
((Q 2))

ED D
NUME PREDICAT:
.Q
NUMAR CLAUZA:
.2
CLAUZA S-A STERS
LIST Q
((Q 1))
((Q I))
((Q 2))

ED M
NUME PREDICAT:
.Q
NUMAR CLAUZA:
.2
((Q I M))
LIST Q
((Q 1))
((Q I M))
((Q 2))
```

Fig.2.

```
((ED X)
  (PP NUME PREDICAT :)
  (R Y)
  (PP NUMAR CLAUZA :)
  (R Z)
  (ED Y Z X))
((ED X Y Z)
  (EQ Z E)
  (CL ((X|x)|y) 1 z)
  (SUM X1 1 Y)
  (SUM z X1 Y1)/
  (CL ((X|Z1)|x1) Y1 y1)
  (RFILL (((X|Z1)|x1)) z1)/
  (ADDCL z1 X1))
((ED X Y Z)
  (EQ Z M)
  (CL ((X|x)|y) 1 z)
  (SUM X1 1 Y)
  (SUM z X1 Y1)/
  (CL ((X|Z1)|x1) Y1 y1)
  (RFILL (((X|Z1)|x1)) z1)/
  (DELCL X Y)
  (ADDCL z1 X1))
((ED X Y Z)
  (EQ Z D)
  (CL ((X|x)|y) 1 z)
  (SUM X1 1 Y)
  (SUM z X1 Y1)/
  (DELCL X Y)
  (P " @MCLAUZA S-A STERS@M"))
((ED X Y Z)
  (EQ Z I)
  (CL ((X|x)|y) 1 z)
  (SUM X1 1 Y)
  (SUM z X1 Y1)/
  (CL ((X|Z1)|x1) Y1 y1)
  (SUM X1 1 z1)/
  (P "@MSCRIETI CLAUZA@M")
  (R X2)
  (ADDCL X2 z1))
```

3.Posibilitati de implementare a ciclurilor de programe in limbajul micro-PROLOG

Comanda TEST 6 (fig.3) lanseaza in executie predicatul SUMA pentru calculul repetat al sumei primelor N numere naturale. Prin executarea acestei comenzi,pe ecran se obtine:

```
SE CALCULEAZA SUMA PRIMELOR N NR.
NATURALE,SFIRSIT:END

N=?.1
   S= 1
N=?.2
   S= 3
N=?.3
   S= 6
N=?.4
   S= 10
N=?.END
```

Fig.3.

```
((TEST 6)
  (P "@MSE CALCULEAZA SUMA PRIME
LOR N NR.NATURALE,SFIRSIT:END")
  (REP)
  (P "@MN=?")
  (R X)
  (NOT END X)
  (SUMA X Y)
  (P " S=" Y)
  (EQ X END))

((REP))
((REP))
  (REP))

((END X)
  (EQ X END)
  ABORT)

((SUMA 0 0))
((SUMA X Y)
  (NOT EQ X 0)
  (SUM X -1 Z)
  (SUMA Z x)
  (SUM x X Y)/)
```

4.Bibliografie

[1] Dinca M.-Manual de utilizare micro-PROLOG
[2] Clocksin,Mellsch,Programming in PROLOG, Springer Verlag, 1984
[3] Turbo PROLOG, BORLAND, 1986
[4] Cretu Vladimir, Structuri de date si tehnici de programare, 1987,curs IPT

DAN VLASIE

Limbajul C implementat pe TIM-S

INTRODUCERE

Limbajul C este un limbaj de programare universal, caracterizat printr-o exprimare concisă, un control modern al fluxului execuţiei, structuri de date şi un bogat set de operatori.

Limbajul C nu este un limbaj de nivel foarte înalt şi nu este specializat pentru un anumit domeniu de aplicaţii. Absenţa restricţiilor şi generalitatea sa îl fac un limbaj mai convenabil şi mai eficient decît multe alte limbaje.

Deşi este un limbaj de nivel relativ scăzut, C este un limbaj agreabil, expresiv, elastic, care se pretează la o gamă largă de programe. C este un limbaj mic şi se învaţă foarte uşor, iar subtilităţile se reţin pe măsură ce experienţa de programare creşte.

IMPLEMENTARE PE TIM-S

Prezentăm în continuare o scurtă descriere a implementării limbajului C pe gama "Spectrum" realizată de firma HISOFT.

Pentru o bună utilizare a compilatorului trebuie să distingem între mai multe regiuni de exploatare:

1. Regim de compilare fără păstrarea codului obiect.

Textul sursă poate fi adus compilatorului spre prelucrare în următoarele moduri:

a) direct, prin editarea liniilor în bufferul compilatorului. Astfel, liniile sursă se compilează imediat ce sînt editate. Practic, acest mod nu se foloseşte decît pentru scrierea unor porţiuni mici din program, căci:

- nu se poate reveni asupra textului deja compilat;
- la apariţia unei erori de compilare se pierde tot codul generat pînă în acel moment;
- după execuţie, atît textul sursă cît şi codul obiect generat se pierd.

b) folosind editarea prealabilă a textului şi comanda §include.

Mai exact, apăsînd CS+1 se apelează editorul de texte integrat (vezi Apendixul), care permite generarea de text sursă, modificări şi corecţii asupra lui, listarea la im-

primantă precum şi citirea/scrierea textului sub forma unui fişier pe/de pe suport magnetic extern (bandă sau microdrive).

Revenind la nivelul compilatorului, prin tipărirea comenzii ǂinclude are loc compilarea textului sursă obţinut în memorie cu ajutorul editorului. La detectarea unei erori sintactice, prin apelarea editorului are loc poziţionarea automată în text la linia ce a produs eroarea. După efectuarea corecţiilor, compilarea trebuie relansată.

Acesta este cel mai uzual mod de lucru.

c) - prin citirea de pe suportul magnetic extern.

Folosind comanda ǂinclude nume-fişier, compilatorul este determinat să citească liniile sursă din fişierul extern nume-fişier, creat în prealabil printr-o salvare din editorul de texte. Se recomandă ca textul păstrat în fişierul extern să fi fost anterior validat de erori sintactice.

Această modalitate de lucru este indicată pentru programele mari care nu încap în zona afectată de editorul de texte, precum şi pentru păstrarea sub formă de bibliotecă sursă a unor funcţii uzuale care să se includă în compilarea unor diferite programe.

În aplicaţii se pot utiliza în conlucrare toate cele 3 modalităţi enumerate.

În momentul în care compilatorul depistează în textul sursă caracterul special EOF, care se obţine tastînd SS+I (valoarea sa în programe este -1), compilarea se încheie, iar codul maşină obţinut se lansează în execuţie. Execuţia are loc efectiv la apăsarea tastei Y. Compilatorul rămîne în memorie şi poate fi folosit într-o nouă aplicaţie.

2. Compilarea şi păstrarea codului obiect

Dacă prima linie din textul sursă este ǂtranslate nume-obiect, atunci la întîlnirea caracterului EOF va avea loc salvarea codului obiect pe suport extern sub forma unui program în cod maşină cu numele nume-obiect.

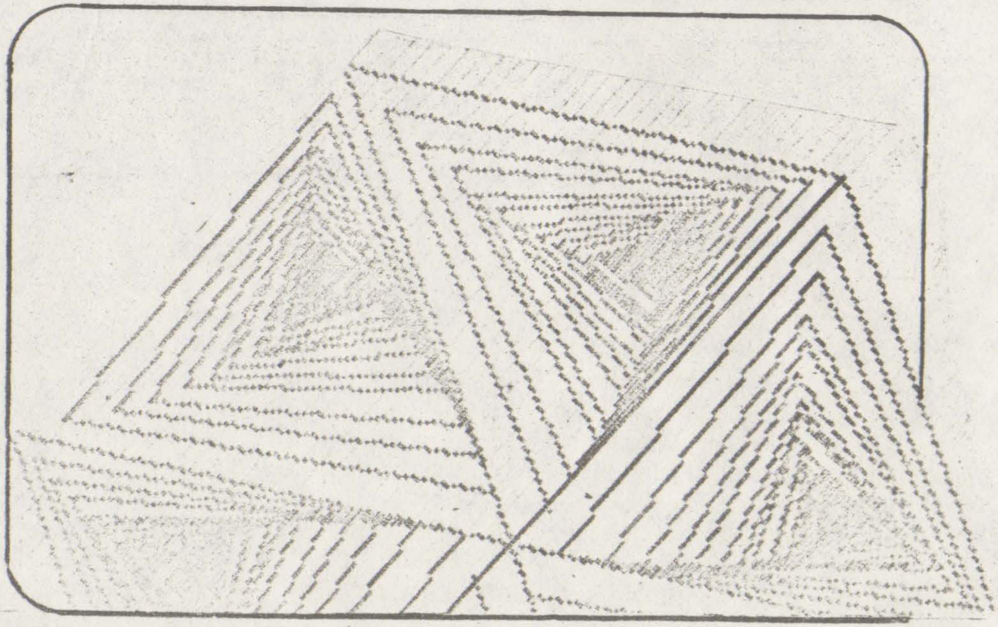
În urma acestei operaţii, compilatorul este distrus, o eventuală utilizarea necesitînd reîncărcarea sa.

3. Regimul interpretativ

Prin tastarea comenzii ǂdirect+, sistemul intră în regim interpretativ. Aceasta înseamnă că orice instrucţie care apare în bufferul compilatorului se execută direct, adică: variabilele pot fi asigurate, ciclurile se pornesc, funcţiile pot fi apelate etc. În regim interpretativ vor fi date doar instrucţii executabile (nu şi definiţii sau declaraţii). Normal, acest mod de lucru este util atunci cînd execuţiile sînt însoţite de scrierea rezultatelor.

Lucrul în regim interpretativ este recomandat în faza de depanare şi punere la punct a programelor.

Prin comanda ǂdirect - se abandonează regimul interpretativ.



Particularități și restricții :

- Conversiile de tip se realizează printr-un operator special: "cast". Sintaxa utilizării lui este:

cast (nume-tip) expresie, unde nume-tip este un tip de bază sau un nume obținut printr-o declarație typedef.

- Nu există implementat tipul flowt sau double. Așadar în aritmetică nu se pot folosi decît întregi cu reprezentare pe 2 octeți.

- Variabilele de tip pointer și variabilele de tip întreg nu se pot asigna între ele, cu excepția valorii întregi Ø care se consideră pointerul nul. Este permisă însă o conversie de tip explicită cu ajutorul operatorului cast înaintea asigurării.

- Nu există comanda ‡define cu parametri.

- Operatorul virgulă nu este implementat.

- Variabilele automatice nu se pot inițializa la declarare.

Despre fișiere :

Un program C poate întreține fișiere pe suport magnetic extern. Ne vom referi în continuare la fișierele de pe casete. Evident, la un moment dat nu poate fi deschis decît un singur fișier pe casetă.

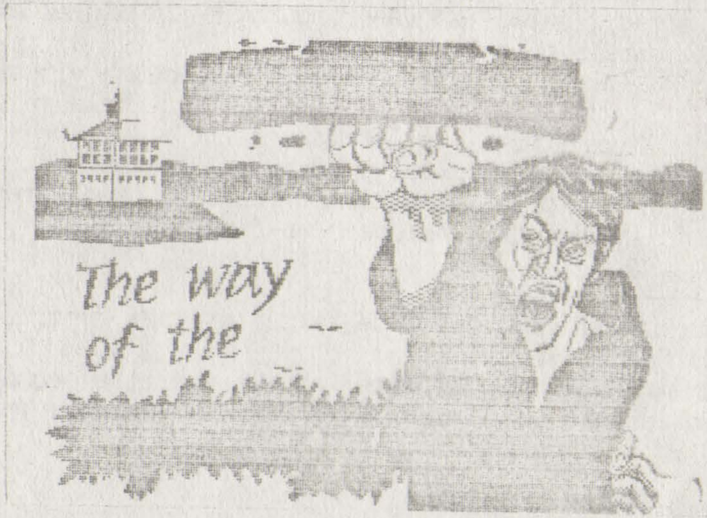
Tipul FILE se poate defini ca fiind ‡define FILE int.

Funcțiile de acces la fișier sînt Fopen(), fclose(), getc(), putc(), și sînt predefinite, adică nu necesită o includere dintr-o bibliotecă.

Toate intrările/ieșirile de/pe casetă se fac la nivel de caracter. Caracterele sînt colectate în blocuri înainte de a fi scrise pe bandă. La fel, de pe bandă se citesc blocuri de caractere.

Înainte de a citi/scrie de pe/pe bandă, este necesar un apel la funcția fopen (nume fis,mod). Dacă modul este scriere, atunci are loc scrierea pe bandă a unui antet cu numele nume-fis. Dacă modul este citire, atunci banda este parcursă pentru a găsi un antet cu numele nume-fis. Apoi funcțiile getc și putc se folosesc pentru a citi sau a scrie caractere, iar în final fclose va scrie și ultimul bloc pe fișierul de ieșire.

Cînd sistemul va scrie pe casetă, culoarea borderului devine roșie pentru circa 5 secunde. În acest moment casetofonul trebuie să meargă în regim RECORD. Cînd sistemul citește de pe casetă, borderul este roșu timp de o secundă, spre a indica pornirea casetofonului în regim PLAY. Cînd programul a citit un bloc, caseta trebuie oprită și repornită cînd e nevoie de citirea unui nou bloc. Compilatorul e capabil să compileze text sursă dintr-un bloc cît timp se desfășoară pauza dintre blocuri, așa că magnetofonul poate merge continuu.



Cîteva considerații de ordin practic

- Deoarece în timpul rulării nu au loc teste suplimentare de corectitudine, programul poate eșua în cele mai neașteptate și catastrofale moduri. În unele cazuri se poate distruge însuși compilatorul. De aceea este indicat de a salva textul sursă înainte de a-l compila și lansa în execuție !

Aveţi mare grijă la :

- folosirea defectuoasă a indicilor în vectori .
- asignarea unei valori greşite unui pointer şi apoi folosirea adresei respective.
- numărul incorect de argumente oferit unei funcţii.
- apelul unei funcţii nedefinite prin intermediul unui pointer la funcţie.
- apelul unei funcţii nedefinite în regim interpretativ.

- Dacă la compilare apare mesajul ERROR 60 LIMIT: no more memory, puteţi să vă "procuraţi" spaţiu suplimentar prin următoarele procedee :

  - daţi comanda ‡error care va cauza eliberarea zonei de memorie ocupată de textele mesajelor de eroare.
  - salvaţi textul în fişiere pe bandă, ştergeţi textul din editor şi folosiţi compilarea cu comanda ‡include nume-fiş (vezi 1.C)

APPENDIX :

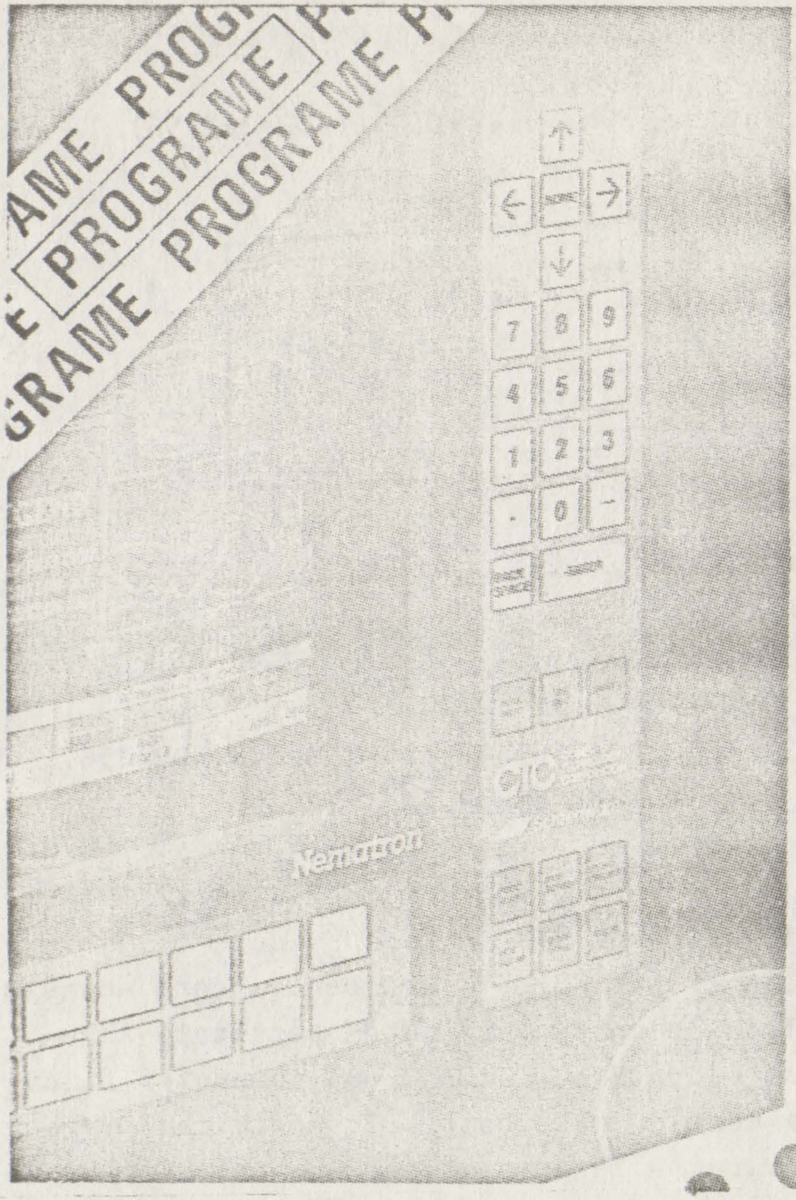
Utilizarea editorului de texte integrat (vezi şi editorul Pascal-Hisoft).

Promptul editorului este ">". La apariţia promptului se pot da următoarele comenzi (listăm doar pe cele mai uzuale) :

>In,m : are drept efect inserarea de text începînd din linia n, raţia liniilor fiind m. Terminarea inserării sevmarchează prin CS+1 la început de linie.
>Dn,m : şterge liniile n÷m inclusiv.
>Ln,m : listează liniile n÷m inclusiv.
>En : permite corectarea liniei n cu ajutorul tastelor:
  CS+5,CS+8 : poziţionare
  I : inserare caractere
  C : schimbare caracter
  K : ştergere caracter
>C : revenirea în compilator.
>Wn,m : listare la imprimantă.
>Pn,m,nume-fiş : salvează pe bandă liniile n÷m, formînd un fişier cu numele nume-fiş.
>G,nume-fiş : aduce de pe bandă în editor textul din fişierul nume-fiş.

Bibliografie :

- "The Programming Language" - Brian Kernighan & Dennis Ritchie - Prentice-Hall, 1978.
- "Programarea în limbajul C - sistemul U" - I.T.C.I., Filiala Cluj-Napoca.
- "Hisoft C" - user guide.



PROF. ŞERBAN MARINEL

compact compact compact screen$ screen$ screen$

Se ştie că un ecran complet (SCREEN$) ocupă 6912 baiţi (6144 informaţii + 768 atribute). De multe ori este necesar ca într-un program să existe mai multe SCREEN - uri, pregătite din timp cu un produs adecvat

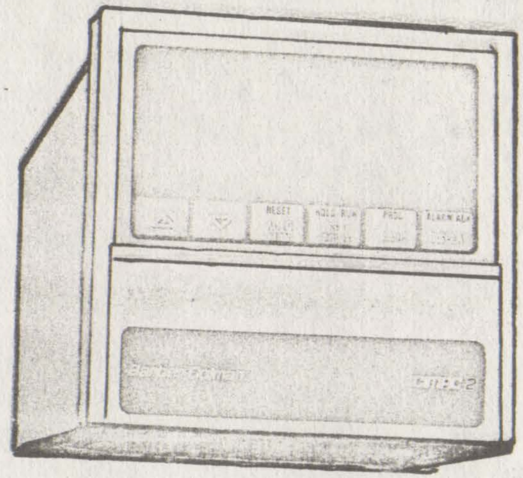
(ART STUDIO, ARTIST, etc.). In acest caz însă spaţiul de memorie ocupat de SCREEN$ - uri ar fi prea mare; de ex. pentru 4 SCREEN$ -uri 27648 baiţi, ceea ce lasă doar aproximativ 13 K liberi pentru programul BASIC. Este necesară deci memorarea acestor SCREEN -uri într-o formă compactă.

Programul COMPACT SCREEN$ realizează această compactare la nivel de octet, contorizînd numărul de octeţi consecutivi identici şi memorînd contorul respectiv şi valoarea octetului. In final se memorează în primii doi octeţi ai zonei numărul de grupe astfel realizat.

Corectitudinea algoritmilor de compactare şi restaurare a fost testată prin realizarea programelor BASIC respective, deoarece compactarea şi restaurarea decurgea lent s-a trecut la realizarea rutinei respective în cod maşină. Rutina de compactare are 160 baiţi iar cea de restaurare 52 baiţi, putînd fi eventual scurtate. Prima a fost asamblată la adresa 65324 iar a doua la 65484.

Pentru realizarea restaurării unui SCREEN compactat, încărcat la adresa ADR secvenţa care trebuie realizată este:

```
POKE 65494,ADR+2-256*INT((ADR+2)/256)
POKE 65495,INT((ADR+2)/256)
```



```
POKE 65498,ADR-256*INT(ADR/256)
POKE 65499,INT(ADR/256)
```

apoi,normal:

RANDOMIZE USR 65484. (vezi programul "ex.decomp")

Obs. Există SCREEN$ -uri pentru care aceasta metoda de compactare nu dă rezultate bune, zona ocupată de SCREEN$ -ul compactat fiind mai mare de 6912 baiţi. In acest caz, evident se renunţă la compactare.

Cele două programe de compactare şi respectiv restaurare sînt prezentate în continuare:

```
 1o CLEAR 65323: BORDER 1: PAPER 2: INK 6: C
LS
 2o GO SUB 2ooo
 3o PRINT AT 1,0; INK 2;: LOAD ""CODE 65324
 4o CLS : GO SUB 2ooo: PRINT AT 4,0; INK 6;
"Pregăteşte caseta cu SCREEN$ -ul"'' pentru
compactat"
1oo LOAD ""SCREEN
12o RANDOMIZE USR 65324
15o LET cit=PEEK 5oooo+256*PEEK 5ooo1
16o LET cit=cit*2+2
17o CLS : GO SUB 2ooo: PRINT '''" Au fost
ocupaţi:";
INVERSE 1;cit; INVERSE o;" octeţi"''"
    adică:";
INVERSE 1;cit/1o24; INVERSE o;" K": PAUSE o
175 IF cit>6912 THEN GO TO 1ooo
176 PRINT ''"     Restaurez SCREEN  -ul?": P
AUSE o
177 IF INKEY ="D" THEN  GO TO 2oo
178 IF INKEY ="N" THEN  GO TO 189
179 IF CODE (INKEY )=7 THEN  LET adr=17o: GO
TO 6oo
18o PAUSE o: GO TO 177
189 PRINT ''"Salvez SCREEN-ul compactat?(Y/N
)": PAUSE o
19o IF INKEY$="Y" THEN  GO TO 4oo
191 IF INKEY$="N" THEN  GO TO 5oo
192 IF CODE (INKEY$)=7 THEN  LET adr=17o: GO
TO 6oo
199 PAUSE o: GO TO 19o
2oo RANDOMIZE USR 65484
21o PAUSE o
22o GO TO 17o
4oo CLS : GO SUB 2ooo: PRINT ''''''''
"Pregăteşte caseta pentru salvat"'
'"Bytes: SCREEN$ COMPACT 5oooo,";cit
41o SAVE "SCREEN$ COMPACT"CODE 5oooo,CIT
5oo CLS : GO SUB 2ooo
51o PRINT ''''''"              Alt SCREEN$ ?"
'"               Y/N": PAUSE o
52o IF INKEY$="Y" THEN  GO TO 4o
53o IF INKEY$="N" THEN  STOP
54o IF CODE (INKEY$)=7 THEN  LET adr=5oo: GO TO 6oo
55o PAUSE o: GO TO 52o
6oo CLS : GO SUB 2ooo
6o5 PRINT AT 5,8; PAPER 1;"
61o FOR i=6 TO 16: PRINT AT i,7; PAPER 6;
"               ";AT i-1,26;
PAPER 1;" ": NEXT i
```

```
6[illegible] AT 7,9; INK 1; PAPER 6;
"MARINEL SERBAN";AT 8,12;"profesor";
AT 11,12;"TIMISOARA"; AT 12,8;"str.HEBE nr.38
/B/7"; AT 14,1o;"tel.961/59964"
 63o PAUSE o: CLS : GO TO adr
1ooo CLS : PRINT ''''''
"          DECOMPACTARE!!!"''''"          ";
cit;
" 6912": PAUSE o : GO TO 5oo
1999 PAUSE o
2ooo PRINT AT o,8; PAPER 5; INK 1; BRIGHT 1;
" COMPACT SCREEN$ ";AT 1,12; PAPER 7; INK 1;
"INFO-1987": RETURN
9999 SAVE "compact" LINE 1: SAVE "compactOBJ"
CODE 65324,211:
 VERIFY "compact": VERIFY "compactOBJ"CODE
```

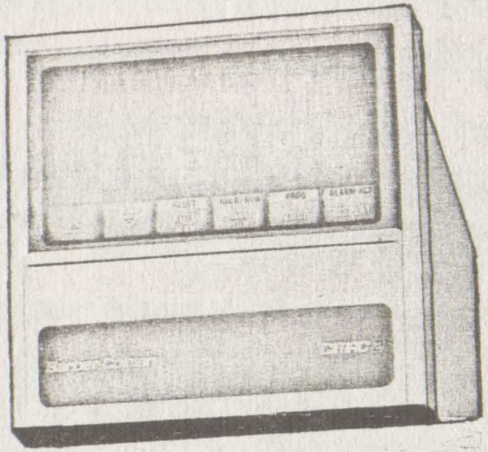
```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: oo
```

```
          5· ▪C-
FF2C   1o          ORG  65324
FF2C   2o          PUSH IX
FF2E   3o          PUSH IY
FF3o   4o          DI
FF31   5o          LD   IY,5ooo2
FF35   6o          LD   IX,16384
FF39   7o          LD   DE,1
FF3C   8o          LD   HL,1
FF3F   9o          LD   A,(IX+o)
FF42   1oo         LD   BC,6911
FF45   11o ALT     CP   (IX+1)
FF48   12o         JR   Z,EGAL
FF4A   13o PUS     PUSH AF
FF4B   14o         LD   A,D
FF4C   15o         CP   o
FF4E   16o         JR   Z,MIC255
FF5o   17o         LD   (IY+o),255
FF54   18o         POP  AF
FF55   19o         LD   (IY+1),A
FF58   2oo         INC  IY
FF5A   21o         INC  IY
FF5C   22o         PUSH BC
FF5D   23o         LD   B,255
FF5F   24o B255    DEC  DE
```

```
FF6o   25o          DJNZ B255
FF62   26o          POP  BC
FF63   27o          INC  HL
FF64   28o          JR   PUS
FF66   29o MIC255   LD   A,E
FF67   3oo          CP   o
FF69   31o          JR   Z,OCTV
FF6B   32o          POP  AF
```



```
FF6C   33o          LD   (IY+o),E
FF6F   34o          LD   (IY+1),A
FF72   35o          INC  HL
FF73   36o          INC  IY
FF75   37o          INC  IY
FF77   38o          JR   LDA
FF79   39o OCTV     POP  AF
FF7A   4oo LDA      LD   A,(IX+1)
FF7D   41o          LD   DE,1
FF8o   42o          JR   NEXTI
FF82   43o EGAL     INC  DE
FF83   44o NEXTI    INC  IX
FF85   45o          DEC  BC
FF86   46o          PUSH AF
FF87   47o          LD   A,B
FF88   48o          OR   C
FF89   49o          CP   o
FF8B   5oo          JR   Z,CONT
FF8D   51o          POP  AF
FF8E   52o          JR   ALT
FF9o   53o CONT     LD   A,D
FF91   54o          CP   o
FF93   55o          JR   Z,MI255
FF95   56o          LD   (IY+o),255
FF99   57o          POP  AF
FF9A   58o          LD   (IY+1),A
FF9D   59o          INC  IY
FF9F   6oo          INC  IY
FFA1   61o          PUSH BC
FFA2   62o          LD   B,255
FFA4   63o DECDE    DEC  DE
```

```
FFA5   64o          DJNZ DECDE
FFA7   65o          POP  BC
FFA8   66o          INC  HL
FFA9   67o          PUSH AF
FFAA   68o          JR   CONT
FFAC   69o MI255    LD   A,E
FFAD   7oo          CP   o
FFAF   71o          JR   Z,OCTV1
FFB1   72o          POP  AF
FFB2   73o          LD   (IY+o),E
FFB5   74o          LD   (IY+1),A
FFB8   75o          INC  HL
FFB9   76o          JR   LDA1
FFBB   77o OCTV1    POP  AF
FFBC   78o LDA1     LD   IX,5oooo
FFCo   79o          LD   (IX+o),L
FFC3   8oo          LD   (IX+1),H
FFC6   81o          EI
FFC7   82o          POP  IY
FFC9   83o          POP  IX
FFCB   84o          RET

Pass 2 errors: oo

Table used:    159  from   227
```

```
   5 CLEAR 6545o: LOAD "decomp"CODE 65484:
LOAD "SCREEN  comp"CODE 40000
  1o PAPER 6: BORDER 5: INK 2: CLS
  2o PRINT AT 4,1o;"DECOMPACTARE"
  3o PRINT AT 6,2;"SCREEN$ -ul compactat la ADR"
  4o PRINT AT 8,o;"POKE 65494,ADR+2-256*INT((
ADR+2)      /256)";AT 1o,o;"POKE 654
95,INT((ADR+2)/256)";AT 11,o;
"POKE 65498,ADR-256*INT(ADR/256)";AT 12,o;
"POKE 654
99,INT(ADR/256)"
  5o PRINT AT 14,1;"Apelare cu RANDOMIZE USR
65484"
  6o PRINT AT 16,o;"EXEMPLU:";AT 17,6;"ADR=4o
ooo";AT 2o,1o;"Apasa o tasta": PAUSE o: LET a
dr=4oooo: GO SUB 1ooo
  7o RANDOMIZE USR 65484: PAUSE o: GO TO 1o
 999 STOP
1ooo POKE 65494,adr+2-256*INT ((adr+2)/256)
1o1o POKE 65495,INT ((adr+2)/256)
1o2o POKE 65498,adr-256*INT (adr/256)
1o3o POKE 65499,INT (adr/256)
1o4o RETURN
9999 SAVE "ex.decomp" LINE 1: SAVE "decomp"CO
DE 65484,52: SAVE "SCREEN  comp"CODE 4oooo,55
```

```
oo: PRINT AT o,o; FLASH 1;" VERIFY ": VERIFY "
ex.decomp": VERIFY "decomp"CODE : VERIFY "SCRE
EN$ comp"CODE
```

```
*HISOFT GENS3M2 ASSEMBLER*
     ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: oo

        1o *C-
FFCC    2o        ORG   65484
FFCC    3o        PUSH  AF
FFCD    4o        PUSH  BC
FFCE    5o        PUSH  DE
FFCF    6o        PUSH  HL
FFDo    7o        PUSH  IX
FFD2    8o        PUSH  IY
FFD4    9o        LD    IX,5ooo2
FFD8   1oo        LD    BC,(5oooo)
FFDC   11o        LD    HL,16384
FFDF   12o        DEC   BC
FFEo   13o INCA1  PUSH  BC
FFE1   14o        LD    B,(IX+o)
FFE4   15o        LD    A,(IX+1)
FFE7   16o INCA   LD    (HL),A
FFE8   17o        INC   HL
FFE9   18o        DJNZ  INCA
FFEB   19o        POP   BC
FFEC   2oo        INC   IX
FFEE   21o        INC   IX
FFFo   22o        DEC   BC
FFF1   23o        LD    A,B
FFF2   24o        OR    C
FFF3   25o        JR    NZ,INCA1
FFF5   26o        POP   IY
FFF7   27o        POP   IX
FFF9   28o        POP   HL
FFFA   29o        POP   DE
FFFB   3oo        POP   BC
FFFC   31o        POP   AF
FFFD   32o        RET

Pass 2 errors: oo

Table used:    36  from   147
```

ING. MIODRAG PUTERITY

program pentru vizualizat sprite-uri, seturi de caractere și u.d.g-uri

0. GENERALITATI

Ati dorit veodata sa vedeti sau sa folositi sprite-urile dintr-un joc ? Dar un set de caractere mai deosebit ? Daca raspunsul e da, aveti cele necesare in programul de mai jos.

N.B. SPRITE - un grup de "caractere grafice" folosite in special la animatie in jocurile video, dar si in programe mai serioase.

Ca orice program, cel prezentat devine pe deplin functional cind are un nume adecvat; in consecinta l-am denumit SPRITE BUSTER.

Programul e scris aproape in intregime in limbaj de asamblare Z80, deci s-ar adresa in special cunoscatorilor (desi urmarind atent instructiunile de mai jos se poate utiliza cu succes si de incepatori).

VIAȚĂ FĂRĂ DE MOARTE ...LA CLAVIATURĂ!

student Mircea TEODORESCU
student Laurenţiu EMIL

Exasperarea jucătorului începător în faţa perspectivei neplăcute de a nu reuşi niciodată să ducă la bun sfîrşit un joc cu un număr de vieţi limitat poate fi ocolită dacă se reuşeşte aflarea adresei la care este stocat numărul de vieţi acordat de programator.

La "HIPERACTION", de pildă, nu se poate interveni direct, programul trebuie analizat cu MONS 3M21. La adresa A9E5H vom găsi, în hexagesimal, numărul vieţilor.

La "SABRE WULF" este mai simplu. Programul se încarcă astfel:

CLEAR 24575:LOAD""SCREEN$:LOAD ""CODE:LOAD""CODE:LOAD""CODE:

LOAD""CODE:POKE43575,89:POKE 45520,89:RANDOMIZE USR 23427

Cu cele două POKE-uri introduse, vieţile nu mai scad.

Alte jocuri, alte adrese:

LA GRANDE FUGA : POKE 56473,167
BARNEY BURGER'S : POKE 59593,195
FRANK : nivele POKE 28275, n
şi POKE 28281, n
vieţi POKE 28287, m

unde n şi m sînt cifrele dorite.

TUTANK : POKE 27783, 0
COMMANDO : POKE 31107, n

unde n reprezintă numărul de vieţi şi grenade.

ZZOOM : POKE 24743, 0

GREEN BERET : se încarcă partea de BASIC cu MONS, după care se înscrie 0 la adresele: A2B3, A2B4, A2B5, A2B6, apoi pune JP 60DB şi se porneşte din nou BASIC-ul programului.

GO TO HELL : POKE 63254, 0
KNIGHTLORE : POKE 53567, 0
PENTAGRAM : POKE 49917, 0
UNDERWULD : POKE 59376,167

1. DESCRIEREA PROGRAMULUI

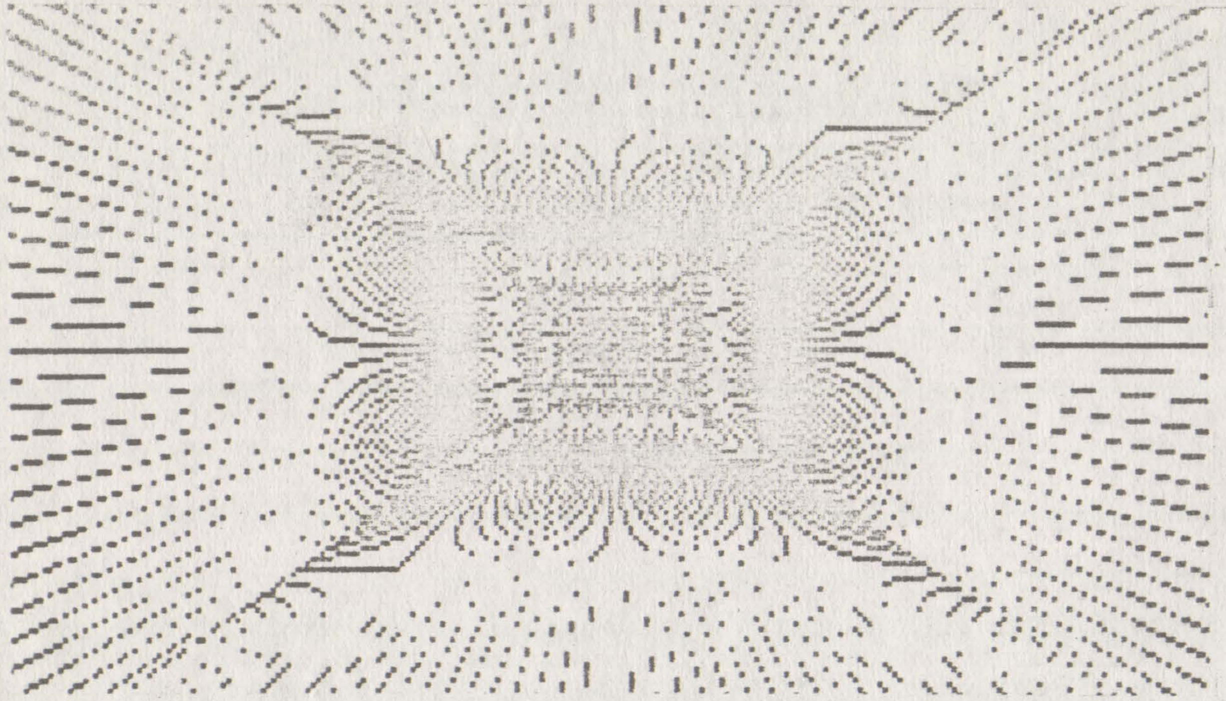
Sprite-urile pot fi afisate in doua moduri (care depind de formatul lor de inmagazinare). Pentru a le intelege, considerati un sprite ca o matrice ale carei elemente sint pixelii sprite-ului. Primul mod de afisare se face preluind octetii de date din memorie si afisindu-i "intii pe coloana". Acest mod se selecteaza cu tasta (1). Evident, al doilea mod, selectat cu tasta (2), va afisa datele "intii pe linie". Ca sa fiu totusi riguros, o sa complic putin lucrurile, spunind ca sistemul "natural" de afisare al Spectrum-ului face ca elementele matricii amintite sa fie de 8 pixeli pe orizontala si 1 pixel pe verticala. Cu alte cuvinte, trecind pe orizontala de la un pixel la altul, raminem de 7 ori in cadrul aceluiasi octet din memoria video, si a 8-a oara trecem in alt octet. Deplasindu-ne pe verticala, cu fiecare pixel trecem in alt octet.

Dimensiunea "ferestrei" de afisare a sprite-ului se poate modifica cu tastele (6,7,8,9) sau cu un joystick in configuratie Sinclair.

Tasta (O) sau butonul de foc sterg zona de afisare. Va trebui folosit atunci cind dimensiunile sprite-ului scad, pentru a curata ecranul de "dire". O versiune ulterioara va inlatura acest neajuns.

Zona de memorie afisata este controlata de un asa numit "memory pointer". Acesta poate fi deplasat prin memorie cu tastele (Q,W,E,R) spre stinga rapid, stinga, dreapta, dreapta rapid (respectiv). Tastele (O,P) scad sau aduna din/la "memory pointer" un numar de octeti egal cu "aria sprite-ului". Daca sprite-urile sint inmagazinate intr-o ordine speciala aceste taste pot fi folosite pentru realizarea animatiei (ex. Manic Miner, Marsport, Cyclone, etc.)

Tasta (L) se foloseste pentru a incarca programul caruia dorim sa-i vedem sprite-urile. Programul se incarca ca un bloc "headerless" de maximum 42239 octeti. Tasta (S) salveaza fereastra de afisare itr-un format de tip SCREEN$, iar tasta (T) salveaza datele sprite-ului in acelasi format ca cel de inmagazinare. O versiune superioara va permite conversia celor doua moduri de afisare. Toate subrutinele amintite in acest paragraf sint protejate la BREAK si la erori de incarcare astfel incit acestea nu vor duce la caderea sistemului.



Tasta (C) se poate folosi pentru copiera zonei de afisare pe o imprimanta de tip ZX Printer. Un BUG cunoscut in aceasta subrutina va fi remediat intr-o versiune superioara.

Puteti folosi un "rastru de bright". Se activeaza cu tasta (G) si se dezactiveaza cu tasta (F).

Toate functiile sint prevazute sa lucreze cu o bucla de intirziere a carei contanta de timp se poate modifica cu tastele (A,Z).

Tasta (B) reseteaza sistemul (ca si USR 0).

O p[illegible] din parametri controlati de program se afiseaza in a ultima [illegible] a ecranului.

2. INSTRUCTIUNI DE ASAMBLARE

Folositi asamblorul GENS3M (sau versiuni superioare) pe care il incarcati suficienţ de jos in memorie pentru a avea loc

ZOTYOCOPY

Traducerea : Dana TÖRÖK

Program de copiere maghiar, unul din programele cele mai răspîndite în Ungaria. Posibilităţile sale sînt mai variate ca la COPIER FM3, dar modul de utilizare necesită mai multă atenţie.

Datele completate pe pagină sînt identice cu semnele folosite la COPIER FM3.

În partea superioară a ecranului se afişează capacitatea de memorie, care în starea de bază este de 41780 byte.

Dedesubt sînt expuse opţiunile programului. Cele utilizabile sînt notate cu alb; alegerea se face prin apăsarea iniţialei cuvîntului ales.

Pentru completarea paginii se întrerupe cu BREAK. Atunci folosim primul set de instrucţiuni :

FORGET - şterge fila umplută, aduce programul la starea de bază.

LOAD - porneşte încărcarea.

VERIFY - verifică în mod corespunzător comanda BASIC, dacă pagina corespunde cu cea din memorie.

END (CAPS-SHIFT+E) - ieşirea din program.

▷▷▷▷▷▷▷▷▷▷▷▷▷

La alegerea "SAVE" intră în funcţie al doilea set de instrucţiuni; prin alegerea "AUTO" salvăm fiecare filă care este în memorie; cu alegerea "HAND" cerem conducerea manuală şi intervine a 3-a categorie de instrucţiuni:

UP - ne plimbăm în sus printre programele înregistrate iar cursorul marchează programul actual.

DOWN - ne plimbăm în jos, iar cursorul indică programul actual.

ENTER - activarea comenzii SAVE/LOAD/VERIFY în cazul cînd capacitatea de memorie nu este suficientă, cu CAPS-SHIFT şi M - ajungem în stadiul de "max-byte" în care putem copia 49076 byte.

Cu acest mod de funcţionare nu se poate însă salva decît o singură dată programul completat, apoi Zotyocopy se opreşte pe comanda BREAK şi se resetează.

La completarea greşită a filei se opreşte încărcarea şi în colţul de sus (locul de capacitate a memoriei) apare indicaţia TAPE ERROR.

Se vehiculează şi programul "Zotyocopy +", care se deosebeşte prin faptul că programele la "max-byte" se pot salva de mai multe ori cu apăsarea pe ENTER.

pentru fisierul text. Personal, am folosit pachetul DEVPAC 7.8 (o reusita "compilare" a lui GENS3M si MONS3M cu un grup de subrutine utile) si desi fisierul text incepea la aproximativ 45000 (zecimal), nu am avut probleme.

Tastati cu atentie textul. Cind ati incheiat, faceti o copie de siguranta pe banda (cu comanda P) si dati comanda de asamblare (A).

La intrebarea asamblorului "Table size:" folositi valoarea estimata de acesta tastind <ENER> ca raspuns, iar la intrebarea urmatoare "Options:" tastati 16 sau 20. Aceste optiuni plaseaza codul obiect asamblat conform directivelor ORG, imediat dupa tabela de simboluri. Pentru a-i afla adresa de inceput, procedati dupa cum urmeaza:

- cu comanda (X) data asamblorului determinati sfirsitul fisierului text (al doilea parametru). Fie aceasta valoare x1.
- la sfirsitul asamblarii veti primi mesajul "Table used <n1> from <n2>. Retineti valoarea n2.
- adresa cautata este x1+n2+2. Fie aceasta valoare adr.

Reveniti in BASIC (B) si tastati:

```
SAVE "un_nume" CODE adr,2048
```

si veti obtine codul obiect care asteapta cu nerabdare sa fie lansat.

Toate cele expuse mai sus sint necesare deoarece, in scopul folosirii la maximum a memoriei disponibile, codul obiect trebuie plasat in cei 2K din "mijlocul" memoriei video. Asamblarea directa in acest loc nu este posibila din motive evidente (incercati asamblarea cu optiunea 0 sau 4).

Din acest moment treburile devin mult mai simple, tastati

```
LOAD "" CODE 18432
RANDOMIZE USR 18432
```

si daca nu ati gresit pe undeva programul va porni. Daca doriti sa-i dati o forma mai eleganta, tastati si loaderul BASIC din listing.

3. RECOMANDARI DE UTILIZARE

La sfirsitul incarcarii unei portiuni de cod, "memory pointer-ul" se afla la adresa 23296. Selectati zona maxima de afisare (DX=32 si DY=64) si un mod oarecare. Deplasati-va (cel

mai rapid cu tastele (O,P)) pina cind veti observa un "model" caracteristic zonei de date pentru sprite-uri (experienta va va ajuta). Actionati tastele (6,7,8,9,0) pentru a gasi dimensiunea sprite-ului cautat, apoi deplasativa incet cu (W,E) pina cind sprite-ul va aparea corect. Puteti sa va ajutati de rastru pentru a stabili dimensiunea sprite-ului. Daca nu gasiti nici un sprite refaceti operatiile de mai sus in modul 2 de afisare.

In timp veti cistiga o indeminare mai mare si puteti micsora valoarea buclei de intirziere.

RECOMANDARE: In modul 1 de afisare conteaza inaltime sprite-ului (DY) iar in modul 2 latimea sa (DX). Folositi-va de aceasta observatie pentru a gasi rapid grupuri de sprite-uri.

Unele jocuri (ex. nemuritorul Manic Miner) au sprite-urile asezate foarte ordonat. Daca gasiti corect dimensiunea sprite-urilor (in exemplul dat DX=2 si DY=16) si pozitionati corect sprite-ul in fereastra, puteti anima sprite-ul (prin afisarea rapida a sprite-urilor conjugate) cu tastele (O,P).

Seturile de caractere si UDG-urile le veti gasi in modul 1.

Un numar foarte mic de jocuri (ex. produsele firmei Mikro Gen) au datele codificate. Programul de mai sus nu poate gasi sprite-uri (poate o versiune superioara). In alte programe (ex. firma A.C.G. - gama ULTIMATE PLAY THE GAME) sprite-urile vor aparea inversate pe verticala si oglindite pe orizontala. O versiune superioara va face posibila afisarea acestora in modul in care ele apar pe ecran.

BIBLIOGRAFIE SELECTIVA

[1] D. Laine, Machine code applications, etc.

[2] D. Webb, Advanced Spectrum machine language, 1983, Melbourne House Publishers

[3] I. Logan, F. O'Hara, The complete Spectrum ROM disassembly, 1983, Melbourne House Publishers

In caz ca aveti orice fel de probleme legate de acest program, iata adresa mea:

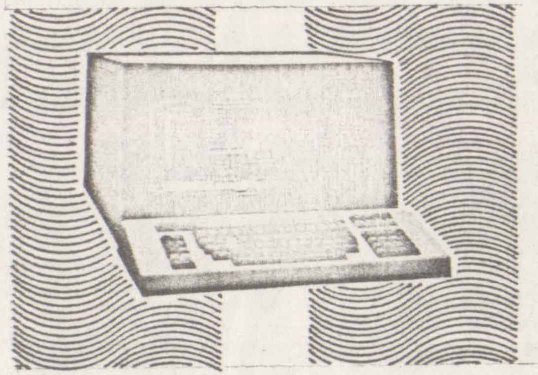
Puterity Miodrag, Str. Vicentiu Babes, Nr. 12,
Arad, 2900, Tel. 966/16109

FRECVENŢMETRU

```
   10 CLEAR 63999
   20 PRINT "Frecvenţmetru": PRINT
   40 FOR A=64000 TO 64055
   50 READ X: POKE A,X
   70 NEXT A
   80 PRINT "Domeniu:50Hz - 20kHz"
   90 PRINT
  100 PRINT "Semnalul se aplică la
intrarea casetofonului şi apoi se
apasă tasta A !"
  110 PRINT
  115 POKE 23658,8
  120 IF INKEY$()"A" THEN GO TO 120
  125 CLS
  130 LET N=USR 64000+65536*PEEK
23670
  140 PRINT AT 10,0;"FRECVENTA:";
INVERSE 1;167492500/(33*N+1400);
INVERSE 0; " Hz "
  150 GO TO 130
  160 LET A$=INKEY$
  170 IF A$()"D" AND A$()"N" THEN
GO TO 160
  180 IF A$="D" THEN PRINT §0;AT
1,0; "                          "
: GO TO 130
  190 STOP
  200 DATA 243,46,0,1,64,50,219,
-2,161,32,-5,219,-2,161,40,-5,85,
93,19,219,-2,161,32,-6,213,85,93,
19,219,-2,161,40,-6,213,16,236,6,
99,-3,112,82,225,175,209,25,20,6,
0,16,-6,50,118,92,68,77,-5,201
 1000 SAVE "FRECV-M" LINE 1
```

```
100 ;************************
110 ;* SPRITE BUSTER MP 1987*
120 ;************************
130 ;
140         ORG    #4800
150         ENT    $
160 ;
170 INIT    DI
180         LD     A,1
190         OUT    (254),A
200         LD     IX,MOD
210         LD     SP,SPWS+31
220 K_SCAN  LD     BC,63486 ;1..5
230         IN     C,(C)
240         BIT    0,C
250         CALL   Z,M1
260         BIT    1,C
270         CALL   Z,M2
280         LD     BC,64510 ;Q..T
290         IN     C,(C)
300         BIT    0,C
310         CALL   Z,FLFT
320         BIT    1,C
330         CALL   Z,LFT
340         BIT    2,C
350         CALL   Z,RIG
360         BIT    3,C
370         CALL   Z,FRIG
380         BIT    4,C
390         CALL   Z,SPSA
400         LD     BC,61438 ;0..6
410         IN     C,(C)
420         BIT    1,C
430         CALL   Z,SU
440         BIT    2,C
450         CALL   Z,SD
460         BIT    3,C
470         CALL   Z,SR
480         BIT    4,C
490         CALL   Z,SL
500         BIT    0,C
510         CALL   Z,CLSW
520         LD     BC,65022 ;A..G
530         IN     C,(C)
540         BIT    0,C
550         CALL   Z,SP_UP
560         BIT    1,C
570         CALL   Z,SAVE
580         BIT    3,C
590         CALL   Z,FILAT
600         BIT    4,C
610         CALL   Z,GRID
620         LD     BC,57342 ;P..Y
630         IN     C,(C)
640         BIT    0,C
650         CALL   Z,ARIG
660         BIT    1,C
670         CALL   Z,ALFT
680         LD     BC,49150 ;EN.H
690         IN     C,(C)
700         BIT    1,C
710         CALL   Z,LOAD
720         LD     BC,32766 ;SP.B
730         IN     C,(C)
740         BIT    4,C
750         CALL   Z,EXIT
760         LD     BC,65278 ;CS.V
770         IN     C,(C)
780         BIT    1,C
790         CALL   Z,SP_DW
800         BIT    3,C
810         CALL   Z,COPY
820         CALL   CALAR
830         CALL   PR_MP
840         CALL   PR_MD
850         CALL   PR_DEL
860         CALL   PR_DXY
870         CALL   PR_AR
880         CALL   PW
890         LD     C,100
900 KS2     LD     B,0
910 KS1     DJNZ   KS1
920         DEC    C
930         JR     NZ,KS2
940         JP     K_SCAN
950 SP_UP   LD     HL,KS2+1
960         DEC    (HL)
970         RET
980 SP_DW   LD     HL,KS2+1
990         INC    (HL)
1000        RET
1010 M1     LD     A,1
1020        JR     M2+2
1030 M2     LD     A,2
1040        LD     (MOD),A
1050        RET
1060 FLFT   LD     HL,(MEMP)
1070        DEC    H
1080        JR     MPRET
1090 LFT    LD     HL,(MEMP)
1100        DEC    HL
1110        JR     MPRET
1120 RIG    LD     HL,(MEMP)
1130        INC    HL
1140        JR     MPRET
1150 FRIG   LD     HL,(MEMP)
1160        INC    H
1170 MPRET  LD     (MEMP),HL
1180        RET
1190 ARIG   LD     DE,(AREA)
1200        LD     HL,(MEMP)
1210        ADD    HL,DE
1220        JR     MPRET
1230 ALFT   LD     DE,(AREA)
1240        LD     HL,(MEMP)
```

```
1250          OR    A
1260          SBC   HL,DE
1270          JR    MFRET
1280 SU       LD    A,(DY)
1290          CP    1
1300          RET   Z
1310          DEC   A
1320          LD    (DY),A
1330          RET
1340 SD       LD    A,(DY)
1350          CP    64
1360          RET   Z
1370          INC   A
1380          LD    (DY),A
1390          RET
1400 SL       LD    A,(DX)
1410          CP    1
1420          RET   Z
1430          DEC   A
1440          LD    (DX),A
1450          RET
1460 SR       LD    A,(DX)
1470          CP    32
1480          RET   Z
1490          INC   A
1500          LD    (DX),A
1510          RET
1520 ;
1530 CLSW     PUSH  BC
1540          LD    HL,16384
1550          LD    BC,2047
1560          LD    (HL),L
1570          LD    D,H
1580          LD    E,1
1590          LDIR
1600          POP   BC
1610          RET
1620 ;
1630 PW       LD    HL,(MEMP)
1640          LD    A,0
1650          LD    (IX+1),A
1660          LD    (IX+2),A
1670          LD    A,(MOD)
1680          CP    2
1690          JP    Z,PW2
1700          LD    B,(IX+3)
1710 NXTX1    LD    (IX+2),0
1720          LD    C,(IX+4)
1730 NXTY1    CALL  DF_ADR
```

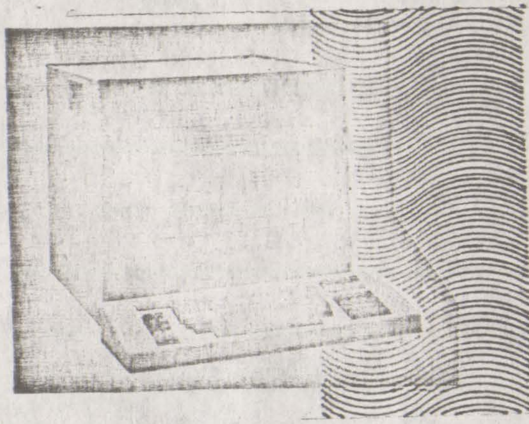


```
1740          LD    A,(HL)
1750          LD    (DE),A
1760          INC   (IX+2)
1770          INC   HL
1780          DEC   C
1790          JP    NZ,NXTY1
1800          INC   (IX+1)
1810          DJNZ  NXTX1
1820          RET
1830 PW2      LD    C,(IX+4)
1840 NXTY2    LD    (IX+1),0
1850          LD    B,(IX+3)
1860 NXTX2    CALL  DF_ADR
1870          LD    A,(HL)
1880          LD    (DE),A
1890          INC   (IX+1)
1900          INC   HL
1910          DJNZ  NXTX2
1920          INC   (IX+2)
1930          DEC   C
1940          JP    NZ,NXTY2
1950          RET
1960 ;
1970 DF_ADR   LD    A,(Y)
1980          AND   7
1990          OR    %01000000
2000          LD    D,A
2010          LD    A,(X)
2020          AND   %00011111
2030          LD    E,A
2040          LD    A,(Y)
2050          SLA   A
2060          SLA   A
2070          AND   %11100000
2080          OR    E
2090          LD    E,A
2100          RET
2110 ;
2120 MOD      DEFB  1
2130 X        DEFB  0
2140 Y        DEFB  0
2150 DX       DEFB  1
2160 DY       DEFB  1
2170 AREA     DEFW  1
2180 MEMP     DEFW  23296
2190 SPWS     DEFS  32
2200 ;
2210 EXIT     JP    0
2220 FILAT    PUSH  BC
2230          LD    HL,22528
2240          LD    D,H
2250          LD    E,1
2260          LD    BC,255
```

```
2270        LD    A,%00110001 ;
INK 1, PAPER 6
2280        LD    (HL),A
2290        LDIR
2300        POP   BC
2310        RET
2320 ;
2330 GRID   PUSH  BC
2340        PUSH  HL
2350        LD    HL,22528
2360        LD    C,4
2370 GRID1  CALL  GP
2380        CALL  GI
2390        DEC   C
2400        JR    NZ,GRID1
2410        POP   HL
2420        POP   BC
2430        RET
2440 GP     LD    B,32 ;L. PARA
2450 GP1    BIT   0,B
2460        JR    NZ,IMPAR1
2470        SET   6,(HL)
2480        INC   HL
2490        DJNZ  GP1
2500        RET
2510 IMPAR1 RES   6,(HL)
2520        INC   HL
2530        DJNZ  GP1
2540        RET
2550 GI     LD    B,32
2560 GI1    BIT   0,B
2570        JR    NZ,IMPAR2
2580        RES   6,(HL)
2590        INC   HL
2600        DJNZ  GI1
2610 IMPAR2 SET   6,(HL)
2620        INC   HL
2630        DJNZ  GI1
2640        RET
2650 ;
2660 LOAD   PUSH  IX
2670        PUSH  BC
2680        LD    IX,#5B00
2690        LD    DE,42239
2700        XOR   A
2710        SCF
2720        EX    AF,AF'
2730        CALL  #0562
2740        JR    LSRET
2750 ;
2760 SAVE   PUSH  IX
2770        PUSH  BC
2780        LD    HL,2048
2790        LD    (HDLEN),HL
2800        LD    IX,HEAD
2810        LD    DE,17
2820        XOR   A
2830        CALL  #04C6
2840        JR    NC,LSRET
2850        LD    IX,16384
2860        LD    DE,2048
2870        LD    A,255
2880        CALL  #04C6
2890        JR    LSRET
2900 ;
2910 SPSA   PUSH  IX
2920        PUSH  BC
2930        LD    HL,(AREA)
2940        LD    (HDLEN),HL
2950        LD    IX,HEAD
2960        LD    DE,17
2970        XOR   A
2980        CALL  #04C6
2990        JR    NC,LSRET
3000        LD    IX,(MEMP)
3010        LD    DE,(AREA)
3020        LD    A,255
3030        CALL  #04C6
3040 LSRET  LD    A,1
3050        OUT   (254),A
3060        POP   BC
3070        POP   IX
3080        RET
3090 ;
3100 HEAD   DEFB  3
3110        DEFM  "MP_SPRITE!"
3120 HDLEN  DEFW  0
3130        DEFW  16384
3140        DEFW  0
3150 ;
3160 HLDEC  LD    DE,10000
3170        LD    BC,0
3180        OR    A
3190 L1     SBC   HL,DE
3200        INC   BC
3210        JP    NC,L1
3220        LD    A,C
3230        LD    (N5),A
3240        ADD   HL,DE
3250        LD    DE,1000
3260        LD    C,0
3270        OR    A
3280 L2     SBC   HL,DE
3290        INC   BC
3300        JP    NC,L2
3310        LD    A,C
3320        LD    (N4),A
3330        ADD   HL,DE
3340        LD    DE,100
3350        LD    C,0
3360        OR    A
3370 L3     SBC   HL,DE
3380        INC   BC
3390        JP    NC,L3
3400        LD    A,C
3410        LD    (N3),A
3420        ADD   HL,DE
```

```
3430          LD    DE,10
3440          LD    C,0
3450          OR    A
3460 L4       SBC   HL,DE
3470          INC   BC
3480          JP    NC,L4
3490          LD    A,C
3500          LD    (N2),A
3510          ADD   HL,DE
3520          LD    DE,1
3530          LD    C,0
3540          OR    A
3550 L5       SBC   HL,DE
3560          INC   BC
3570          JP    NC,L5
3580          LD    A,C
3590          LD    (N1),A
3600          RET
3610 ;
3620 N5       DEFB  0
3630 N4       DEFB  0
3640 N3       DEFB  0
3650 N2       DEFB  0
3660 N1       DEFB  0
3670 ;
3680 BASE     DEFW  #3C00
3690 DFCC     DEFW  #4000
3700 ATT      DEFB  %00000111
3710 MASK     DEFB  0
3720 ;
3730 PRINT1   LD    L,A
3740          LD    H,0
3750          ADD   HL,HL
3760          ADD   HL,HL
3770          ADD   HL,HL
3780          LD    DE,(BASE)
3790          ADD   HL,DE
3800          LD    DE,(DFCC)
3810          LD    B,8
3820 NXTROW   LD    A,(HL)
3830          LD    (DE),A
3840          INC   HL
3850          INC   D
3860          DJNZ  NXTROW
3870          LD    A,D
3880          RRCA
3890          RRCA
3900          RRCA
3910          DEC   A
3920          AND   3
3930          OR    #58
3940          LD    D,A
3950          LD    HL,(ATT)
3960          LD    A,(DE)
3970          XOR   L
3980          AND   H
3990          XOR   L
4000          LD    (DE),A
4010          LD    HL,DFCC
4020          INC   (HL)
4030          RET   NZ
4040          INC   HL
4050          LD    A,(HL)
4060          ADD   A,8
4070          LD    (HL),A
4080          RET
4090 ;
4100 LOCATE   LD    A,B
4110          AND   #18
4120          LD    H,A
4130          SET   6,H
4140          RRCA
4150          RRCA
4160          RRCA
4170          OR    #58
4180          LD    D,A
4190          LD    A,B
4200          AND   7
4210          RRCA
4220          RRCA
4230          RRCA
4240          ADD   A,C
4250          LD    L,A
4260          LD    E,A
4270          LD    A,(DE)
4280          LD    (DFCC),HL
4290          RET
4300 ;
4310 PR_MP    PUSH  AF
4320          PUSH  BC
4330          PUSH  DE
4340          PUSH  HL
4350          LD    HL,(MEMP)
4360          CALL  HLDEC
4370          LD    B,22 ;LINE
4380          LD    C,7 ;COL.
4390          CALL  LOCATE
4400          LD    HL,N5
4410          LD    C,5
4420 PMP1     LD    A,(HL)
4430          ADD   A,47
4440          PUSH  HL
4450          CALL  PRINT1
4460          POP   HL
4470          INC   HL
4480          DEC   C
4490          JR    NZ,PMP1
4500          POP   HL
4510          POP   DE
4520          POP   BC
4530          POP   AF
4540          RET
4550 ;
4560 PR_MD    PUSH  AF
4570          PUSH  BC
4580          PUSH  DE
4590          PUSH  HL
```

```
4600          LD   B,17
4610          LD   C,5
4620          CALL LOCATE
4630          LD   A,(MOD)
4640          ADD  A,48
4650          CALL PRINT1
4660          POP  HL
4670          POP  DE
4680          POP  BC
4690          POP  AF
4700          RET
4710 ;
4720 PR_DXY   PUSH AF
4730          PUSH BC
4740          PUSH DE
4750          PUSH HL
4760          LD   A,(DX)
4770          LD   L,A
4780          LD   H,0
4790          CALL HLDEC
4800          LD   B,19
4810          LD   C,3
4820          CALL LOCATE
4830          LD   A,(N2)
4840          ADD  A,47
4850          CALL PRINT1
4860          LD   A,(N1)
4870          ADD  A,47
4880          CALL PRINT1
4890          LD   A,(DY)
4900          LD   L,A
4910          LD   H,0
4920          CALL HLDEC
4930          LD   B,18
4940          LD   C,3
4950          CALL LOCATE
4960          LD   A,(N2)
4970          ADD  A,47
4980          CALL PRINT1
4990          LD   A,(N1)
5000          ADD  A,47
5010          CALL PRINT1
5020          POP  HL
5030          POP  DE
5040          POP  BC
5050          POP  AF
5060          RET
```



```
5070 ;
5080 PR_DEL   PUSH AF
5090          PUSH BC
5100          PUSH DE
5110          PUSH HL
5120          LD   A,(KS2+1)
5130          LD   L,A
5140          LD   H,0
5150          CALL HLDEC
5160          LD   B,21
5170          LD   C,6
5180          CALL LOCATE
5190          LD   C,3
5200          LD   HL,N3
5210 PRDL1    LD   A,(HL)
5220          ADD  A,47
5230          PUSH HL
5240          CALL PRINT1
5250          POP  HL
5260          INC  HL
5270          DEC  C
5280          JR   NZ,PRDL1
5290          POP  HL
5300          POP  DE
5310          POP  BC
5320          POP  AF
5330          RET
5340 ;
5350 PR_AR    PUSH AF
5360          PUSH BC
5370          PUSH DE
5380          PUSH HL
5390          LD   HL,(AREA)
5400          CALL HLDEC
5410          LD   B,20
5420          LD   C,5
5430          CALL LOCATE
5440          LD   HL,N4
5450          LD   C,4
5460 PRAR1    LD   A,(HL)
5470          ADD  A,47
5480          PUSH HL
5490          CALL PRINT1
5500          POP  HL
5510          INC  HL
5520          DEC  C
5530          JR   NZ,PRAR1
5540          POP  HL
5550          POP  DE
5560          POP  BC
5570          POP  AF
5580          RET
5590 ;
5600 CALAR    LD   HL,0
5610          LD   A,(DY)
```

```
5620          LD    E,A
5630          LD    D,0
5640          LD    B,(IX+3) ;DX
5650 AR1      ADD   HL,DE
5660          DJNZ  AR1
5670          LD    (AREA),HL
5680          RET
5690 ;
5700 COPY     LD    HL,16384
5710          LD    C,8
5720 CY2      PUSH  HL
5730          LD    B,8
5740 CY1      PUSH  HL
5750          PUSH  BC
5760          CALL  CROW
5770          POP   BC
5780          POP   HL
5790          INC   H
5800          DJNZ  CY1
5810          POP   HL
5820          LD    DE,32
5830          ADD   HL,DE
5840          DEC   C
5850          JR    NZ,CY2
5860          LD    A,4
5870          OUT   (#FB),A
5880          RET
5890 ;
5900 CROW     XOR   A
5910          OUT   (#FB),A
5920          LD    D,A
5930 CY3      IN    A,(#FB)
5940          ADD   A,A
5950          RET   M
5960          JR    NC,CY3
5970          LD    C,32
5980 CY4      LD    E,(HL)
5990          INC   HL
6000          LD    B,8
6010 CY5      RL    D
6020          RL    E
6030          RR    D
6040 CY6      IN    A,(#FB)
6050          RRA
6060          JR    NC,CY6
6070          LD    A,D
6080          OUT   (#FB),A
6090          DJNZ  CY5
6100          DEC   C
6110          JR    NZ,CY4
6120          RET
```

```
10 REM Miodrag Puterity 1987
20 LOAD ""CODE 18432
30 PRINT AT 17,0;"MODE";AT 18,0;"DY:";AT 19,0;"DX:";AT 20,0;"AREA:";AT 21,0;"D ELAY:";#0;AT 0,0;"MEMORY:"
40 RANDOMIZE USR 18432
50 CLEAR : SAVE CHR$ 22+CHR$ 1+CHR$ 0+"SPR.BUSTR$" LINE 0
```



ING. CONSTANTIN COZMIUC

TIM-S / TIM-S / TIM-
HC-85 / HC-85 / HC-
SPECTRUM / SPEC

SFATURI UTILE

Puteti afla cit spatiu disponibil aveti in memoria calculatorului cu ajutorul instructiunii:

```
PRINT 65535-USR7962;" octeti liberi"
```

Lungimea unui program se poate afla cu:

```
PRINT (PEEK 23627+256*PEEK 23628)-
      (PEEK 23692+256*PEEK 23636)
```

Prin schimbarea valorii variabilei de sistem SCR CT se poate obtine editarea continua, evitind astfel mesajul "scroll ?":
POKE 23692,X unde X este numarul liniilor de editat. Daca X=255 editarea se face continuu.

Sunetul scos de claviatura este controlat de catre variabila de sistem PIP. Valoarea ei initiala este 0, daca se modifica la 128, sunetul devine mai audibil:
POKE 23609,128

Intervalul de repetitie a tastei, controlat de variabila REPDEL, initializata la 35, este de circa 0.7 sec. Se poate modifica astfel:
POKE 23561,20 micsoreaza intervalul la 0.4sec
Daca se introduce valoarea 75 intervalul se mareste la 1.5 sec., iar 0 suprima repetitiile.

Intervalul dintre repetitii este initial de 0.1 sec. Prin POKE 23562,2 intervalul se micsoreaza la 0.04 sec, iar la valoarea 15 devine 0.3 sec.

Schimbarea modului de scriere (K sau L) sepoate face in program, prin modificarea variabilei FLAGS 2:
POKE 23658,8 trece la scierea cu litere mari
POKE 23658,0 trece in modul L

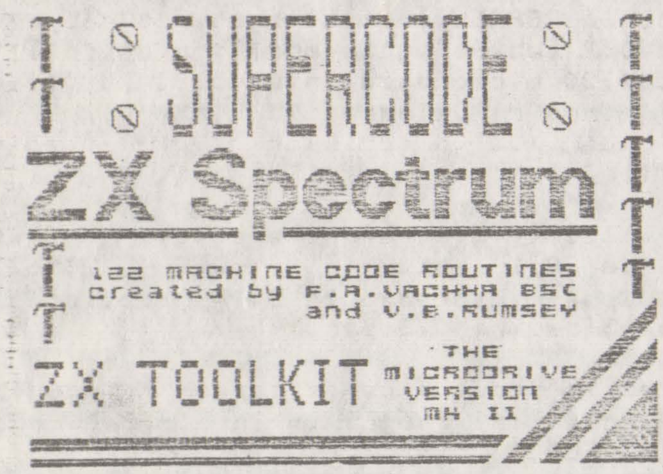
Pentru a va asigura ca numele dvs.nu va fi sters din program ,scrieti :
1 REM copyright 1988 by ....
apoi dati comanda:
POKE 23756,0 si ENTER
in urma careia linia 1 va deveni linia 0, carenu mai poate fi editata, ori stearsa.

Alta cale recomandata pentru evitarea stergerii primei linii este:

```
LET X=PEEK 23635+256*PEEK 23636:POKE X,0:POKE X+1,0
```

Alte linii, daca nu sint apelate cu GO TO, GOSUB, ori RESTORE, pot fi protejate impotriva stergerii, daca li se cunoaste adresa X prin:

```
POKE X-4,0:POKE X-3,0
```



Ne putem asigura impotriva unei opriri a programului prin STOP ori BREAK, daca modificam valoarea variabilei ERR SP, care da adresa mesajului de eroare:

```
POKE 23613,0
```

Plasind in locuri diferite aceasta instructiune, la orice incercare de oprire programul se distruge.

Protejarea unui program se mai poate face si prin alterarea setului de caractere, introducind diferite numere la adresele 23606 si 23607. Normalizarea este necesara inainte de fiecare PRINT si se face prin POKE 0 la prima adresa si 60 la a doua.

Daca prima linie a unui program este REM urmata de:

```
POKE (PEEK 23635+256*PEEK 23636),100
```

atunci programul se poate porni, dar nu se poate lista, pina cind nu inlocuiti valoarea de la adresa de mai sus cu 0.

O alta cale de a afecta listarea este:

```
POKE 23636,150
```

Intoarcerea la listarea normala se poate face cu:

```
POKE 23636,92
```

Un mic program util pentru renumerotarea liniilor:

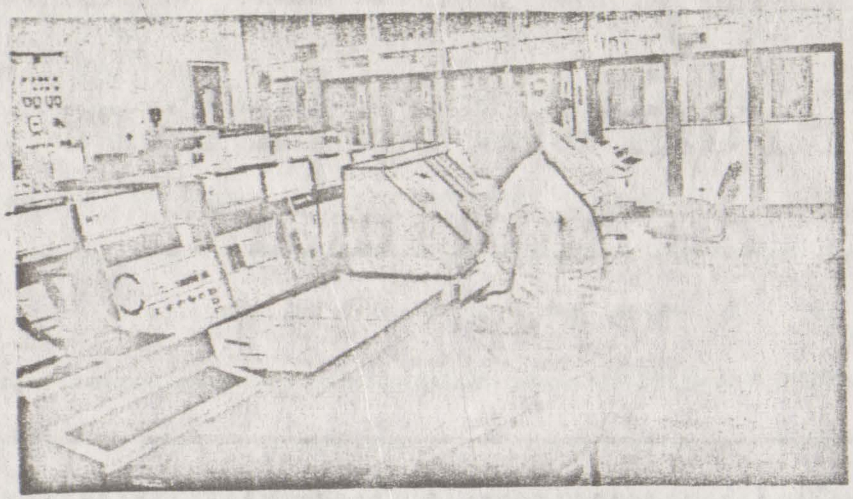
```
9990 REM program renumerotare
9991 INPUT "Numarul liniei de inceput:";d'"
          Pasul:";i
9992 LET a=PEEK 233635+256*PEEK 23636
9993 IF PEEK (a+1)+256*PEEK a =9990 THEN STOP
9994 POKE a,INT d/256
9995 POKE a+1,d-256*INT(d/256)
9996 LET d=d+1
9997 LET a=4+a+PEEK(a+2)+256*PEEK(a-3)
9999 GO TO 9993
```

RANDOMIZE USR 3438 are ca efect stergerea liniilor 22 si 23, similar cu comanda INPUT""

Pentru a incerca sa introduceti mesajul si variabila in orice parte a ecranului, incercati:

```
INPUT AT 22,0;AT X,Y;"Mesaj optional";variab.
```

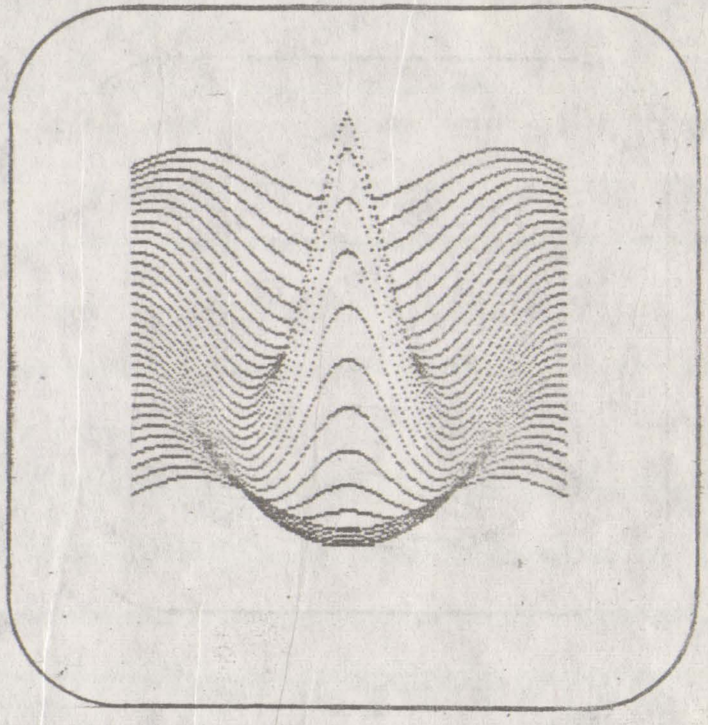
Tiparirea in ultimele doua linii se face cu:
PRINT #1;AT k,0 unde k=0 pentru linia 22 si 1 pentru 23.



Putem contoriza timpul de executie a unei faze din program,daca ne folosim de variabila de sistem FRAMES,și avem grijă să nu folosim RANDOMIZE intre initializare si citire:

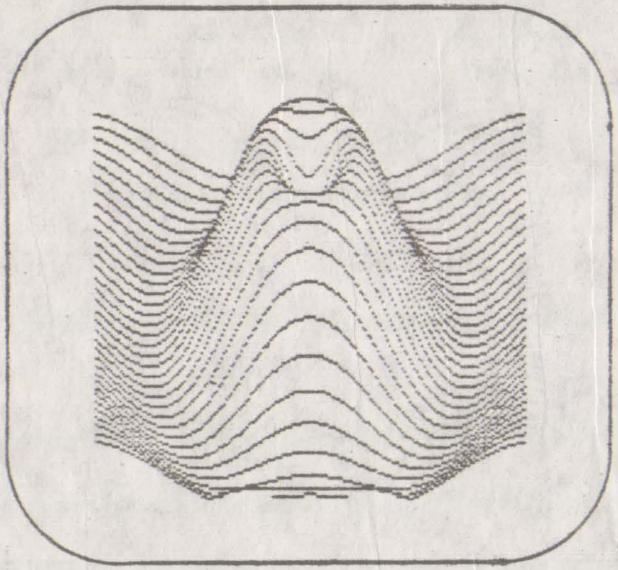
```
 POKE 23672,0:POKE 23673,0:POKE 23674,0
 (Initializare)
      :
      :
 LET  t=PEEK   23672+256*PEEK23673+65536*PEEK
 23674
 LET t1=PEEK   23672+256*PEEK23673+65536*PEEK
 23674
 IF t<t1 THEN LET t=t1
 PRINT "Timp:";INT(t/50);" sec"
 (Citire si afisare )
```

În incheiere,un scurt program pentru grafica tridimensionala:



```
 100 FOR x=40 TO 215
 110 LET b=999: LET t=0
 120 FOR y=16 TO 144 STEP 4
 130 LET r=SQR ((x-127)*(x-127)+
(y-80)*(y-80))/15
 140 LET z=INT (y+90*EXP (-r/3)*
COS r)
```

```
150 IF z<b OR z>t THEN PLOT x,z
160 IF z<b THEN LET b=z
170 IF z>t THEN LET t=z
180 NEXT y
190 NEXT x
```



BIBLIOGRAFIE:

J.F.Sehan *"Clefs pour le ZX Spectrum et Timex 2000"* Ed.du PSI,Paris 1983

Spectrum machine language for the absolute beginner" 1982 Beam Software

[illegible]ckers *"ZX Spectrum"* 1982 Sinclair

Mike Lord *Exploring SPECTRUM BASIC* Timedata Ltd. London 1982

F.R.Vachhr,V.B.Rumsey *SUPERCODE ZX Spectrum* The microdrive version

RADU DRAGOMIR

Program de listare pe imprimanta ROMOM

```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

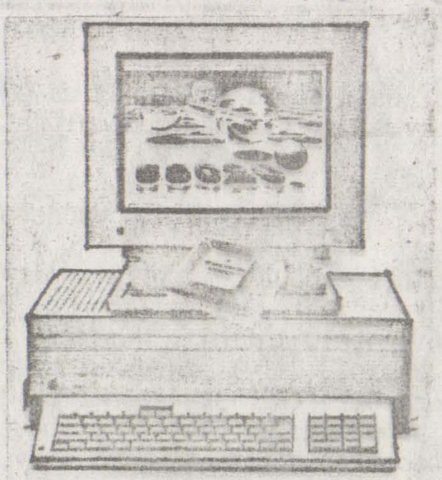
Pass 1 errors: 00

      10 *C-
      20 ;PROGRAM DE LISTARE
      30 ;PE IMPRIMANTA ROMOM
      40 ;DIN BETABASIC 3.1
      50
      60
```

```
         70 ;C 1988 RADU DRAGOMIR
         80
         90
        100
        110 ;AVANTAJE:
        120
        130 ;-ESTE RELOCATABIL
        140 ;-FOLOSESTE INTREAGA
        150 ;LUNGIME A RINDULUI
        160 ;-SE POATE SPECIFICA
        170 ;MARGINEA STINGA CU
        180 ;POKE X+50,N:LPRINT
        190 ;X ADR DE INCARCARE
        200 ;N NR DE SPATII
        210 ;-SE LANSEAZA CU
        220 ;RANDOMIZE USR X
        230 ;********************
        240
C2DE    250          PUSH BC
C2DF    260          LD   A,#E4
C2E1    270          CALL #38D4
C2E4    280          LD   A,#AD
C2E6    290          CALL #38D4
C2E9    300          LD   A,#FF
C2EB    310          CALL #38D4
C2EE    320          POP  BC
C2EF    330          LD   HL,#0019
C2F2    340          ADD  HL,BC
C2F3    350          LD   (#5CC5),H
L
C2F6    360          RET
        370
C2F7    380          CP   #80
C2F9    390          JR   NC,ET3
C2FB    400          CP   #31
C2FD    410          JR   C,ET1
C2FF    420 ET2      CPL
C300    430          JP   #38D4
C303    440 ET1      CP   #0D
C305    450          RET  NZ
C306    460          CPL
C307    470          CALL #38D4
C30A    480          LD   A,#F5
C30C    490          CALL #38D4
C30F    500          LD   B,6
C311    510 REL      PUSH BC
C312    520          LD   A,#DF
C314    530          CALL #38D4
C317    540          POP  BC
C318    550          DJNZ REL
C31A    560          RET
C31B    570 ET3      SUB  #A5
C31D    580          JP   NC,#0C10
C320    590          ADD  A,#A5
C322    600          JP   #FBE4
        610
        620 ;********************

Pass 2 errors: 00

Table used:    53  from   202
```



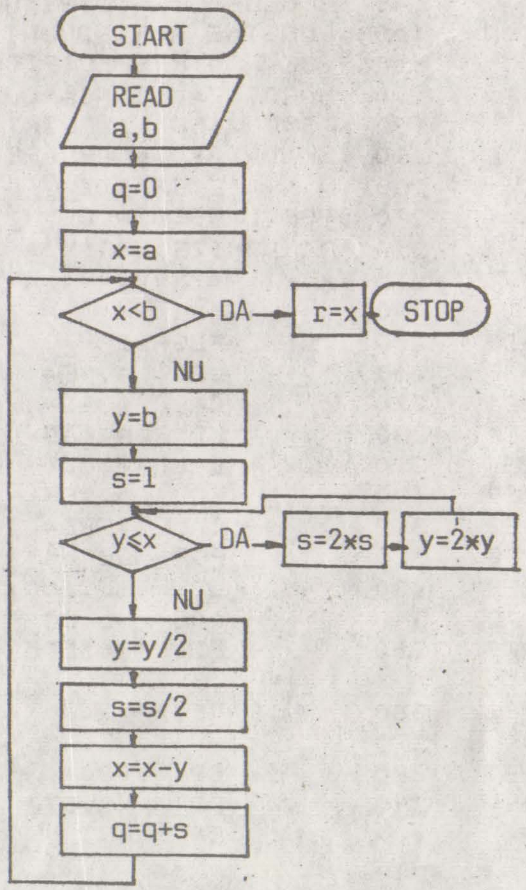
KOOS REMUS

împărțire rapidă

Programul este scris în limbaj de asamblare și se bazează pe faptul că orice număr zecimal se poate scrie sub forma :

$$x = 2^n b_n + 2^{n-1} b_{n-1} + \ldots + 2b_1 + b_0 \quad \text{unde } b_i \in \{0,1\} \quad i=\overline{1,h}$$

Sînt împărțite două numere întregi A și B puse în locațiile (DEIMP) respectiv (IMP) ; B ≠ 0. Dacă B = 0, programul intră în ciclu infinit. Numerele întregi sînt reprezentate pe 2 biți. Cîtul și restul se culeg din locațiile (CIT) respectiv (REST). Are performanțe de viteză superioare algortimului clasic cu scăderi succesive. Are la bază următorul algoritm :



Programul în limbaj de asamblare este :

```
a ------ (DEIMP)
b ------ (IMP)
q ------ (CIT)
x,r ---- (REST)
s ------ DE
y ------ BC

DEIMP   EQU  23296
IMP     EQU  23298
CIT     EQU  23300
REST    EQU  23302
A-PE-B  PUSH AF
        PUSH BC
        PUSH DE
        PUSH HL
GO      LD   HL,0
        LD   (CIT),HL     ; q=0
        LD   HL,(DEIMP)
        LD   (REST),HL    ; x=a
L01     LD   HL,(REST)
        PUSH BC
        LD   BC,(IMP)
        OR   A
        SBC  HL,BC
        POP  BC
        YR   C,END        ; x<b  => ret.
        LD   BC,(IMP)     ; y=b
        LD   DE,1
        LD   HL,(REST)
L02     OR   A
        SBC  HL,BC
        ADD  HL,BC
        YR   C,L03        ; x<y => L03
        SLA  E
        RL   D            ; s=2×s
        SLA  C
        RL   B            ; y=2×y
        YR   C,L04
        YR   L02
L03     OR   A
L04     RR   B
        RR   C
        RR   D            ; y=y/2
        RR   E            ; s=s/2
        OR   A
        LD   HL,(REST)
        SBC  HL,BC        ; x=x-y
        LD   (REST),HL
        LD   HL,(CIT)
        ADD  HL,DE        ; q=q+s
        LD   (CIT),HL
        YR   L01
END     POP  HL
        POP  DE
        POP  BC
        POP  AF
        RET
```



ING. HARALD SCHRIMPF

program de sortare în cod mașină

Acest program de sortare , scris in intregime in cod masina Z80 , a fost realizat in primul rind pentru a obtine timpi acceptabili in sortarea fisierelor. Sper ca veti ramine placut [illegible] de sporul de viteza pe [illegible] veti obtine inlocuind rutinele dv. de sortare scrise in BASIC cu aceasta rutina.

Programul este relocatabil si ocupa un spatiu de doar 150 octeti.

Incarcarea programului o veti face cu:

```
LOAD "SORTmc CODE " CODE xxxxx sau cu
LOAD "" CODE xxxxx
```

Lansarea programului se va face cu:

```
RANDOMIZE USR xxxxx
```

Inainte de a lansa programul in executie va trebui sa-i transmiteti urmatorii parametrii:

- AFIS - adresa de octet a inceputului de fisier
- NRART - numarul de articole din fisier
- LART - lungimea in octeti a unui articol din fisier
- ACIMP - adresa cimpului dupa care se face sortarea fata de inceputul articolului (tot in octeti)
- LCIMP - lumgimea cimpului dupa care se face sortarea

Primii parametrii se vor da pe 2 octeti iar ultimul pe un octet, acestia dind urmatoarele limite programului:

- numarul maxim de articole din fisier - 65535
- lungimea maxima a unui articol -65535 octeti
- lungimea maxima a unui cimp - 255 octeti.

Sint convins ca aceste limite (teoretice) nu vor fi atinse de nici un fisier.

Pentru a memora o variabila "v" pe 2 octeti la adresa "a" se va folosi:

```
POKE a,v-256*INT(v/256)     octetul low
POKE a+1,INT(v/256)         octetul high
```

Initial adresele la care se vor memora parametrii sint urmatoarele:

```
AFIS  - 16384 ($4000)
NRART - 16386 ($4002)
LART  - 16388 ($4004)
ACIMP - 16390 ($4006)
LCIMP - 16392 ($4008)
```

O zona de lungime (LART) incepind de la adresa 16393 ($4009) va fi folosita pentru interschimbarea a 2 articole.

Dupa cum ati observat acest spatiu este inceputul zonei ecran; deci intre transmiterea parametrilor si lansarea programului nu veti mai afisa nimic pe ecran. Un numar de (LART)+9 octeti vor fi afectati din zona ecran dar acest lucru poate fi facut transparent pentru utilizator prin INK=PAPER si CLS inaintea transmiterii parametrilor. Bineinteles ca puteti sa si modificati toate aceste adrese.

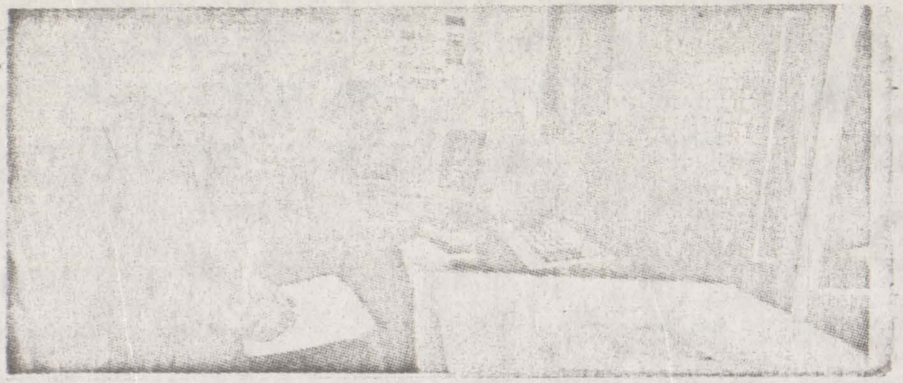
Prezentul program va sorta articolele dupa o singura cheie dar apelindu-l repetat veti putea face sortarea dupa mai multe chei; prima sortare o veti face dupa cea mai putin semnificativa cheie iar ultima dupa cea mai semnificativa cheie.

Sortarea se face in sensul crescator al chei dar cu POKE xxxxx+80,56 se va face in sens descrescator; revenirea la prima varianta se obtine cu POKE xxxxx+80,48.

```
;*************************
;*  Program de sortare  *
;*    in cod masina     *
;*--------------------- *
;*Autor: HARALD SCHRIMPF*
;*      Modern Timisoara*
;*       Dorobantilor 48*
;*        Tel. 31036/185*
;*************************
;                                Locatiile in care se vor
;                                memora urmatoarele:
AFIS    EQU   $4000     ;        Adr. efectiva a fisierului
NRART   EQU   $4002     ;        Numarul de articole
LART    EQU   $4004     ;        Lungimea unui articol
ACIMP   EQU   $4006     ;        Adr.cimpului dupa care se
;                                face sortarea fata de
;                                inceputul articolului
LCIMP   EQU   $4008     ;        Lungimea acestui cimp
BUFFER  EQU   $4009     ;        Zona de lungimea (LART)
;                                pentru interschimbarea a
;                                doua articole
SORT    DI    ;                  Dezactiveaza intreruperile
        PUSH  IX        ;        Salveaza registrele in
        PUSH  IY        ;        stiva
        PUSH  HL
        PUSH  DE
        PUSH  BC
        PUSH  AF
```

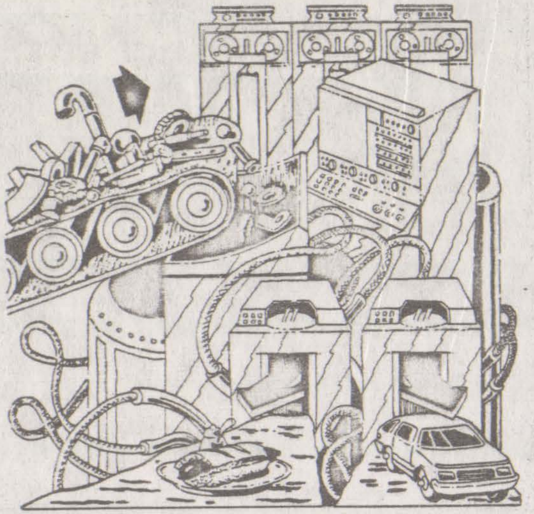
```
        JR   START      ;    Salt la start sortare
RETURN  POP  AF         ;    Reface registrele
        POP  BC
        POP  DE
        POP  HL
        POP  IY
        POP  IX
        EI   ;               Reactiveaza intreruperile
        RET  ;               Intoarcere la programul
;                            apelant
START   LD   HL,(AFIS);      Incarca adresa fisierului
        PUSH HL
        LD   HL,(NRART);     si numarul de articole
        PUSH HL
NEXTA   POP  BC         ;    Nr.articole inca de ordo-
;                            nat+1
        POP  DE         ;    Adresa de unde incepind
;                            fisierul e inca nesortat
        DEC  BC
        LD   A,B        ;    Daca nu mai sint articole
        OR   C          ;    de ordonat
        JR   Z,RETURN   ;    salt la RETURN
        LD   HL,(LART);      Calculeaza adresa urmato-
        ADD  HL,DE      ;    rului articol (i) si
        PUSH HL         ;    salveaza in stiva aceasta
;                            adresa cit si numarul de
        PUSH BC         ;    articole inca de ordonat
        EX   DE,HL      ;    Calculeaza adresa artico-
        LD   HL,(LART);      lului (i-1)
        LD   A,E
        [illegible] L
[illegible]
[illegible]
        [illegible],H
        LD   H,A
        PUSH HL         ;    Salveaza in stiva adresele
        PUSH DE         ;    celor doua articole
```

```
CONT1   POP  IY         ;    Incarca adresele celor 2
        POP  IX         ;    articole de comparat
        LD   BC,(ACIMP);     Aduna la acestea adresa
        ADD  IX,BC      ;    cimpului (in cadrul arti-
        ADD  IY,BC      ;    colului) dupa care se face
;                            sortarea
        LD   A,(LCIMP);      Incarca lungimea acestui
        LD   B,A        ;    cimp si
CONT2   PUSH BC         ;    salveaza in stiva
        LD   A,(IY+0)   ;    Incarca un caracter din
;                            articolul i
        CP   (IX+0)     ;    si compara cu caracterul
;                            corespondent din articolul
;                            i-1
        JR   Z,NEXTCH   ;    Salt la egalitate la urma-
;                            torul caracter
        POP  BC
        JR   NC,NEXTA   ;    Salt la urmatorul articol
;                            daca cele 2 sint ordonate
        PUSH HL         ;    altfel salveaza adresele
        PUSH DE         ;    acestora si
        LD   DE,BUFFER;      schimba pozitia lor
        LD   BC,(LART)
        LDIR ;               Buffer=articol (i-1)
        POP  HL
        POP  DE
        LD   BC,(LART)
        PUSH DE
        PUSH HL
        LDIR ;               Articol (i-1)=articol i
        POP  DE
        LD   HL,BUFFER
        LD   BC,(LART)
        LDIR ;               Articol i=buffer
        POP  DE
        LD   HL,(LART);      Calculeaza adresa artico-
        LD   A,E        ;    lului (i-2)
```



```
        SUB  L
        LD   L,A
        LD   A,D
        SBC  A,H
        LD   H,A
        PUSH HL       ;  Salveaza adresele urmatoa-
        PUSH DE       ;  relor 2 articole de compa-
;                        rat ( i-1 si i-2 )
        LD   BC,(AFIS);  Verifica daca articolul
        LD   A,E      ;  (i-1) nu este primul arti-
        SUB  C        ;  col din fisier
        LD   C,A
        LD   A,D
        SBC  A,B
        OR   C
        JR   NZ,CONT1 ;  Daca nu - salt la CONT1
        POP  BC       ;  daca da - reface pozitia
        POP  BC       ;  stivei
        JR   NEXTA    ;  Salt la ordonarea urmato-
;                        rului articol
NEXTCH  INC  IX       ;  Paseste acum la urmatorul
        INC  IY       ;  caracter din cimp si salt
        POP  BC       ;  la CONT2 daca nu s-a ajuns
        DJNZ CONT2    ;  la sfirsitul cimpului
        JR   NEXTA    ;  daca da - salt la urmato-
;                        rul articol
```

```
5B00  F3      DI
5B01  DDE5    PUSH IX
5B03  FDE5    PUSH IY
5B05  E5      PUSH HL
5B06  D5      PUSH DE
5B07  C5      PUSH BC
5B08  F5      PUSH AF
5B09  180A    JR   §5B15
5B0B  F1      POP  AF
5B0C  C1      POP  BC
5B0D  D1      POP  DE
5B0E  E1      POP  HL
5B0F  FDE1    POP  IY
5B11  DDE1    POP  IX
5B13  FB      EI
5B14  C9      RET
5B15  2A0040  LD   HL,(§4000)
5B18  E5      PUSH HL
5B19  2A0240  LD   HL,(§4002)
5B1C  E5      PUSH HL
5B1D  C1      POP  BC
5B1E  D1      POP  DE
5B1F  0B      DEC  BC
5B20  78      LD   A,B
5B21  B1      OR   C
5B22  28E7    JR   Z,L5B0B
5B24  2A0440  LD   HL,(§4004)
5B27  19      ADD  HL,DE
5B28  E5      PUSH HL
5B29  C5      PUSH BC
5B2A  EB      EX   DE,HL
5B2B  2A0440  LD   HL,(§4004)
5B2E  7B      LD   A,E
5B2F  95      SUB  L
5B30  6F      LD   L,A
5B31  7A      LD   A,D
5B32  9C      SBC  A,H
5B33  67      LD   H,A
5B34  E5      PUSH HL
```



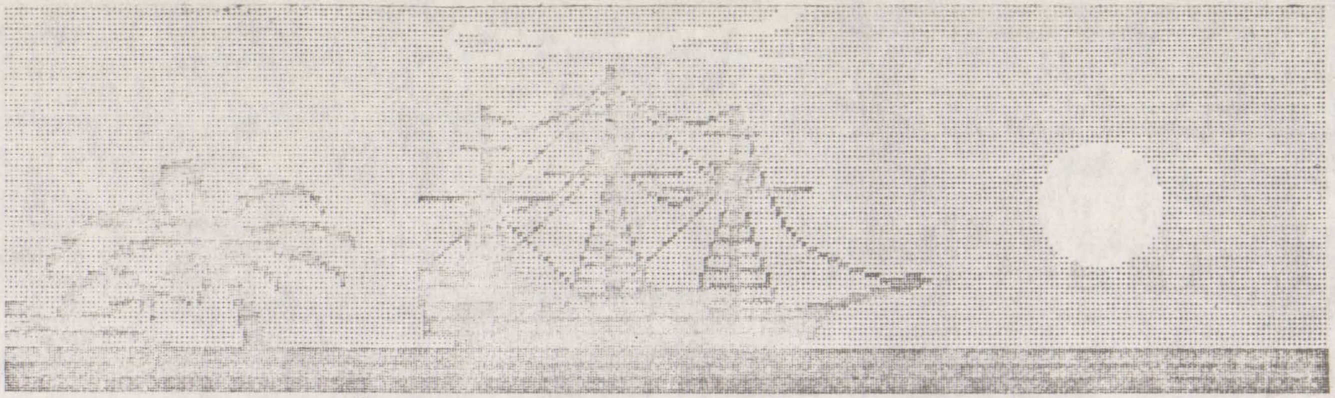
```
5B35  D5        PUSH DE
5B36  FDE1      POP  IY
5B38  DDE1      POP  IX
5B3A  ED4B0640  LD   BC,(§4006)
5B3E  DD09      ADD  IX,BC
5B40  FD09      ADD  IY,BC
5B42  3A0840    LD   A,(§4008)
5B45  47        LD   B,A
5B46  C5        PUSH BC
5B47  FD7E00    LD   A,(IY+0)
5B4A  DDBE00    CP   (IX+0)
5B4D  283E      JR   Z,§5B8D
5B4F  C1        POP  BC
[illegible] 30CB      JR   NC,L5B1D
[illegible] [illegible] PUSH HL
5B53  D5        PUSH DE
5B54  110940    LD   DE,§4009
5B57  ED4B0440  LD   BC,(§4004)
```

```
5B5B  EDB0      LDIR
5B5D  E1        POP  HL
5B5E  D1        POP  DE
5B5F  ED4B0440  LD   BC,(§4004)
5B63  D5        PUSH DE
5B64  E5        PUSH HL
5B65  EDB0      LDIR
5B67  D1        POP  DE
5B68  210940    LD   HL,§4009
5B6B  ED4B0440  LD   BC,(§4004)
5B6F  EDB0      LDIR
5B71  D1        POP  DE
5B72  2A0440    LD   HL,(§4004)
5B75  7B        LD   A,E
5B76  95        SUB  L
5B77  6F        LD   L,A
5B78  7A        LD   A,D
5B79  9C        SBC  A,H
5B7A  67        LD   H,A
5B7B  E5        PUSH HL
5B7C  D5        PUSH DE
5B7D  ED4B0040  LD   BC,(§4000)
5B81  7B        LD   A,E
5B82  91        SUB  C
5B83  4F        LD   C,A
5B84  7A        LD   A,D
5B85  98        SBC  A,B
5B86  B1        OR   C
5B87  20AD      JR   NZ,L5B36
5B89  C1        POP  BC
5B8A  C1        POP  BC
5B8B  1890      JR   L5B1D
5B8D  DD23      INC  IX
5B8F  FD23      INC  IY
5B91  C1        POP  BC
5B92  10B2      DJNZ L5B46
5B94  1887      JR   L5B1D
```

Program demonstrativ pentru imprimanta ROMOM

```
  1)REM DEMONSTRATIE
 50 LET C1=0
    LET C2=2
    LET C3=0
    GO SUB 9190
    GO SUB 9160
    GO SUB 9040
    GO SUB 9100
    LPRINT "DEMONSTRATIE"
    GO SUB 9060
100 LET C1=0
    LET C2=1
    LET C3=2
    GO SUB 9170
101 LET C2=0
    GO SUB 9180
105 GO SUB 9001
    GO SUB 9001
    GO SUB 9040
    LPRINT "Caractere late"
110 GO SUB 9040
    LPRINT "Caractere late"
115 GO SUB 9040
    LPRINT "Caractere late"
120 GO SUB 9020
    GO SUB 9001
    GO SUB 9001
    GO SUB 9050
    LPRINT "Caractere oblice"
121 GO SUB 9050
    GO SUB 9010
    LPRINT "Caractere oblice"
122 GO SUB 9050
    GO SUB 9010
    LPRINT "Caractere oblice"
123 GO SUB 9000
    GO SUB 9000
125 GO SUB 9060
    GO SUB 9070
    LPRINT "Caractere normale -
80 CPI"
127 GO SUB 9080
    LPRINT "Caractere normale -
100 CPI"
129 GO SUB 9090
    LPRINT "Caractere normale -
120 CPI"
130 GO SUB 9040
    GO SUB 9070
    LPRINT "Caractere late - 80
CPI"
132 GO SUB 9040
    GO SUB 9080
    LPRINT "Caractere late - 10
0 CPI"
134 GO SUB 9040
    GO SUB 9090
    LPRINT "Caractere late - 12
0 CPI"
140 GO SUB 9050
    GO SUB 9070
    LPRINT "Caractere oblice -
80 CPI"
142 GO SUB 9050
    GO SUB 9080
    GO SUB 9010
    LPRINT "Caractere oblice -
100 CPI"
144 GO SUB 9050
    GO SUB 9090
    GO SUB 9010
    LPRINT "Caractere oblice -
120 CPI"
145 GO SUB 9040
    GO SUB 9070
    GO SUB 9100
    LPRINT "Aceasta a fost demo
nstratia"
    GO SUB 9060
150 GO SUB 9001
    GO SUB 9040
    LPRINT "LA REVEDERE !"
    GO SUB 9030
    GO SUB 9200
152 GO SUB 9020
    GO SUB 9040
    GO SUB 9070
    LPRINT " ROM-BASIC3"
    GO SUB 9020
    GO SUB 9060
    GO SUB 9090
    LLIST
155 STOP
9000 REM CODURI SPECIALE ROMOM
9001 REM
9002 REM CR=INTOARCERE CAR
9003 LET A=114
     GO SUB 9900
9004 RETURN
9010 REM BS=PAS INAPOI
9011 LET A=123
     GO SUB 9900
9012 RETURN
9020 REM LF=AVANSEAZA UN RIND
```

```
9021 LET A=117
     GO SUB 9900
9022 RETURN
9030 REM FF=SALT LA PAGINA NOUA
9031 LET A=115
     GO SUB 9900
9032 RETURN
9040 REM BDE=CARACTERE LATE
9041 LET A=100
     GO SUB 9900
9042 LET A=36
     GO SUB 9900
9043 LET A=78
     GO SUB 9900
9044 LET A=18
     GO SUB 9900
9045 RETURN
9050 REM SDE=CARACTERE OBLICE
9051 LET A=100
     GO SUB 9900
9052 LET A=36
     GO SUB 9900
9053 LET A=76
     GO SUB 9900
9054 LET A=18
     GO SUB 9900
9055 RETURN
9060 REM NDE=CARACTERE NORMALE
9061 LET A=100
     GO SUB 9900
9062 LET A=36
     GO SUB 9900
9063 LET A=[illegible]
     GO SUB [illegible]
9064 LET A=[illegible]
     GO SUB 9900
9065 RETURN
9070 REM CPI80=80 CARACTERE-RIND
9071 LET A=100
```



```
     GO SUB 9900
9072 LET A=36
     GO SUB 9900
9073 LET A=79
     GO SUB 9900
9074 LET A=95
     GO SUB 9900
9075 LET A=52
     GO SUB 9900
9076 RETURN
9080 REM CPI100=100 CARACTERE-RI
ND
9081 LET A=100
     GO SUB 9900
9082 LET A=36
     GO SUB 9900
9083 LET A=77
     GO SUB 9900
9084 LET A=95
     GO SUB 9900
9085 LET A=52
     GO SUB 9900
9086 RETURN
9090 REM CPI120=120 CARACTERE-RI
ND
9091 LET A=100
     GO SUB 9900
9092 LET A=36
     GO SUB 9900
9093 LET A=76
     GO SUB 9900
9094 LET A=95
     GO SUB 9900
9095 LET A=52
     GO SUB 9900
9096 RETURN
9100 REM UDL=SUBLINIERE
9101 LET A=100
     GO SUB 9900
9102 LET A=36
     GO SUB 9900
9103 LET A=75
     GO SUB 9900
9104 LET A=18
     GO SUB 9900
9105 RETURN
9110 REM LLFC=STERGE MARKERUL DE
 SFIRSIT DE FORMULAR
9111 LET A=100
     GO SUB 9900
9112 LET A=79
     GO SUB 9900
9113 RETURN
9120 REM LLFS=PUNE MARKERUL DE S
FIRSIT DE FORMULAR
9121 GO SUB 9930
9126 LET A=5
     GO SUB 9900
9127 RETURN
9130 REM LPF=FIXEAZA LUNGIMEA FO
RMULARULUI
9131 GO SUB 9930
9136 LET A=2
     GO SUB 9900
9137 RETURN
9140 REM HPRV=POZITIONARE RELATI
VA INAINTE PE RINDUL CURENT
9141 GO SUB 9930
9142 LET A=30
```

```
      GO SUB 9900
9143 RETURN
9150 REM HPRR=POZITIONARE RELATI
VA INAPOI PE RINDUL CURENT
9151 GO SUB 9930
9152 LET A=14
      GO SUB 9900
9153 RETURN
9160 REM HPA=POZITIONARE ABSOLUT
A PE RINDUL CURENT
9161 GO SUB 9930
9162 LET A=31
      GO SUB 9900
9163 RETURN
9170 REM VPRV=POZITIONARE RELATI
VA INAINTE PE VERTICALA
9171 GO SUB 9930
9172 LET A=26
      GO SUB 9900
9173 RETURN
9180 REM VPRR=POZITIONARE RELATI
VA INAPOI PE VERTICALA
9181 GO SUB 9930
9182 LET A=10
      GO SUB 9900
9183 RETURN
9190 REM VPA=POZITIONARE ABSOLUT
A PE VERTICALA
9191 GO SUB 9930
9192 LET A=27
      GO SUB 9900
9193 RETURN
9200 REM DEL=RESETEAZA IMPRIMANT
A
9201 LET A=0
      GO SUB 9900
9202 RETURN
9900 REM PROGRAMUL EFECTIV
9901 OUT 226,A
9902 OUT 226,A+128
9903 OUT 226,A
9905 LET C=IN 224
9906 IF C<=128 THEN GO TO 9905
9907 RETURN
9930 REM PROGRAM AUXILIAR
9931 LET A=100
      GO SUB 9900
9932 LET A=36
      GO SUB 9900
9933 LET A=79-C1
      GO SUB 9900
9934 LET A=79-C2
      GO SUB 9900
9935 LET A=79-C3
      GO SUB 9900
9936 RETURN
9990 SAVE "ROM-BASIC"
      STOP
```

<u>DEMONSTRATIE</u>

Caractere late
Caractere late
Caractere late

Caractere oblice
Caractere oblice
Caractere oblice

Caractere normale - 80 CPI
Caractere normale - 100 CPI
Caractere normale - 120 CPI
Caractere late - 80 CPI
Caractere late - 100 CPI
Caractere late - 120 CPI
Caractere oblice - 80 CPI
Caractere oblice - 100 CPI
Caractere oblice - 120 CPI
<u>Aceasta a fost demonstratia</u>
LA REVEDERE !



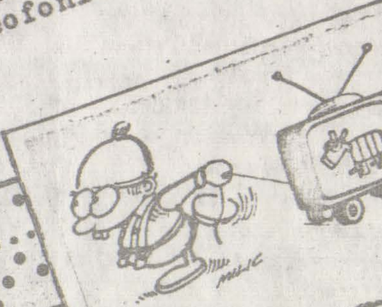
OVIDIU ANDRĂŞESCU

teme de casă

SE DA: Un calculator capabil de grafica (fina), color.
SE CERE: Sa se realizeze desene animate in relief, stereoscopice, dupa urmatorul principiu: o imagine de culoare verde pentru un ochi si o imagine de culoare rosie pentru celalalt ochi. Cu ajutorul unor ochelari bicolori (rosu pentru un ochi si verde pentru celalalt) se vizualizeaza imaginile corespunzatoare celor doi ochi. Prin suprapunerea celor doua imagini la nivelul creierului se creeaza iluzia stereoscopica. Pentru detaliere si calcule a se consulta "Anaglife geometrice", Editura Didactica si Pedagogica, anul 1972.

SE DA: Un calculator capabil de grafica color, o sursa de muzica disco (sau alt gen).
SE CERE: Sa se realizeze o orga de lumini prelucrind semnalul audio in asa fel incit la frecvente diferite sa avem nuante diferite de culori, pe tot ecranul. In felul acesta se elimina inconvenientul principal al orgilor de lumini clasice: trei surse distincte de lumina si culoare si cele trei umbre corespunzatoare lor.

SE DA: Un calculator care are 2-3 generatoare de sunet independente si eventual inca putin hard manufacturat.
SE CERE: Sa se realizeze efecte stereofonice (eventual combinate cu efecte stereoscopice).



AȘTEPTĂM SUGESTIILE DUMNEAVOASTRĂ
ȘI MATERIALELE PROPUSE SPRE PUBLICARE, LA ADRESA:

CASA UNIVERSITARILOR
str. Paris nr. 1
1900 Timișoara

INF nr. 1/1988

inF

Lei 30